-Concepte si aplicatii in vederea artificiala-

-Facial Detector-

-Hadirca Dionisie-

Despre Proiect

Tinta proiectului este de a implementa un algoritm si a antrena un model astfel incat acesta sa ne detecteze fetele din desenul animat “Life with Louie” si sa recunoasca in prealabil pentru fiecare imagine propusa fetele personajelor principale: “Andy, Louie, Ora, Tommy”.

Eu mi-am propus sa urmez conceptele de Computer Vision clasice, astfel incat am folosit:

Task 1, Facial Detection

metoda Slinding Window
Histograme de gradienti orientati (HOG) pentru extragerea caracteristicilor
Masini Vector Suport (SVM) pentru detectia fetelor

Task 2, Facial Recognition

Histograme de gradienti orientati (HOG) pentru extragerea caracteristicilor
Masini Vector Suport (SVM) pentru clasificarea personajelor

Librarii Folosite:

- Numpy
- Scikit-learn
- Matplotlib
- Glob
- Cv2
- Pdb
- Ntpath
- Copy
- Timeit
- Scikit-image
- Os

Pentru toate librariile am folosit ultima versiune

Implementare

Antrenare

Parametri finali utilizati in antrenarea modelului pentru detectie:

```
dim_hog_cell = 8
dim_window = 72
overlap = 0.3
number_positive_examples = 7236
number_negative_examples = 20000
has_annotations = False
threshold = 0
cells_block = (3,3)
orientations=22

pixels_cell = (dim_hog_cell,dim_hog_cell)
resize_factors = [1,1]

resize_param = (int(dim_window*resize_factors[0]),int(dim_window*resize_factors[1]))

iterate_factors = [[1,0.8],[0.9,1]]

for elem in iterate_factors:
```

Unde in esenta, parametri care produc cea mai mare influenta asupra scorului ii reprezinta:

- Dim_hog_cell: numarul de pixeli in interiorul unei celule hog
- Dim_window: dimensiunea ferestrei glisante
- Cells_block: numarul de celule HOG per block care sa fie luate in considerare la detectie
- Iterate_factors, resize_factors, resize_param, sunt utilizati pentru a computa diferite forme ale ferestrei glisante
  Unde practic pentru fiecare element din factori fereastra glisanta scaleaza conform numerelor inregistrate, spre exemplu pentru [1,0.8] vom avea o fereastra dreptunghiulara cu lungimea mai mica decat latimea.
  Acest lucru a fost implementat deoarece, fetele unor personajelor din viata cu louie nu au forma ideala patrata

Rularea Programului:

Pentru obtinerea fisierelor solutie este necesar sa rulati fisierul Hadirca_Dionisie_331.ipynb din folderul “sauce” toate celulele, incepand cu celula 2

Tunarea Parametrilor

Tunarea parametrilor mi s-a parut cea mai grea parte intrucat parametrii utilizati nu functioneaza individual, ci fiecare depinde unul de altul, astfel ca era nevoie de mult trial and error pentru ai gasi pe cei potriviti

Pentru inceput am luat din programul propus la laborator implementarea cu

- Dim_hog_cell = 6, dim-window = 36, cells_block=(2,2) si fetele patrate (aka iterate_factors = [[1,1]]

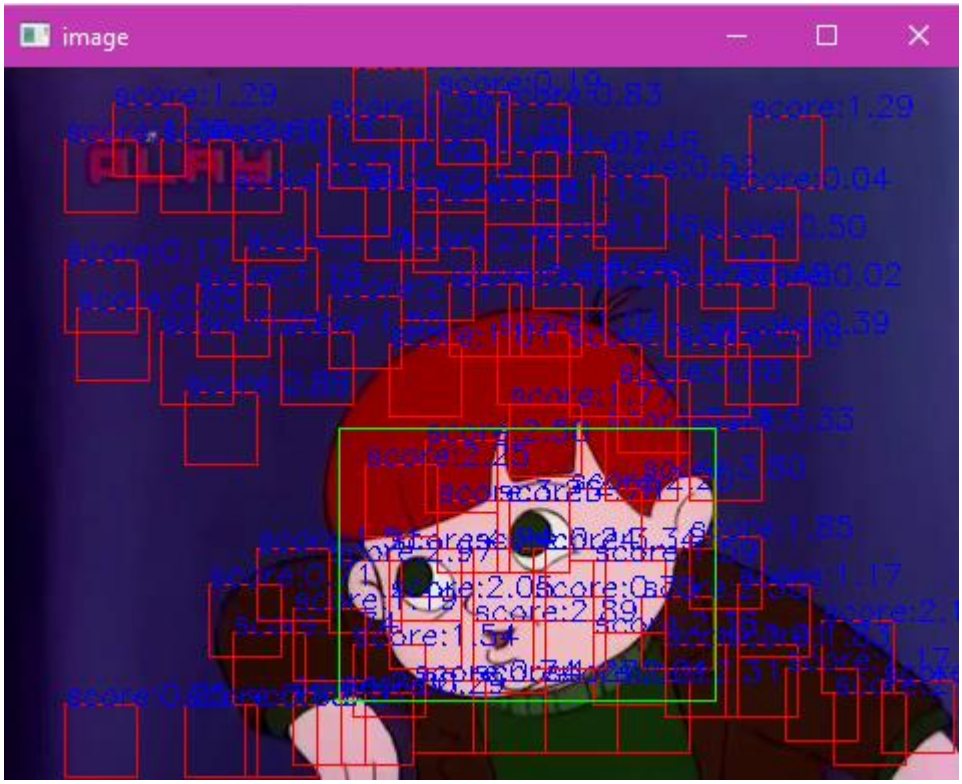
Si am antrenat cu exemple negative alese aleator din poze, neasociate desenului animat (din viata reala)

Pentru aceasta implementare am obtinut urmatorul rezultat:

Average precision 0.012

Ceea ce mi s-a parut, nu prea bine.

Am analizat detectiile furnizate de algoritm pe imagine:



Si din nou am inteles, ca nu e destul de ok pentru a considera proiectul finisat

Am luat urmatoarele notite:

- Este importanta manevrarea cu dimensiunile ferestrei glisante, sau imaginilor
- Pot fi importante caracteristicilor negative
- Pot fi imbunatatiti parametrii pentru extragerea caracteristicilor

In primul rand m-am ocupat de extragerea caracteristicilor negative din imaginile de antrenare propuse.

Caracteristicile sunt extrase in felul urmator: pentru fiecare imagine propusa, se iau aleator o multime de coordonate care reprezinta forme asemanatoare cu fereastra glisanta, si pentru fiecare patch ales se verifica IOU cu fetele din imagine, daca acest IOU depaseste un anumit threshold, patchul nu va fi ales ca exemplu negativ.

Procedura se repeta pana in momentul in care vom avea pentru fiecare imagine propusa cate 5 exemple negative

Dupa implementarea extragerii exemplelor negative am mai executat o rulare, in care am obtinut un average precision de 0.12, un rezultat de 10 ori mai bun decat la inceput

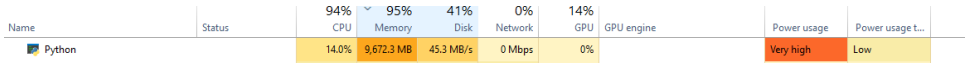
Urmatoarele zile, am decis sa ma ocup de tunarea parametrilor si sa las procedura de multiscaling pentru mai tarziu, astfel a urmat o serie de trial and error, unde setam parametri si observam cum este imaginea, si descriptorii influentati de acestia.

Se intampla sa nu imi ajunga RAM

```
2 train_labels = np.concatenate((np.ones(positive_features.shape[0]), np.zeros(negative_features.sh
3 facial_detector.train_classifier(training_examples, train_labels)

File <__array_function__ internals>:180, in concatenate(*args, **kwargs)

MemoryError: Unable to allocate 3.06 GiB for an array with shape (21596, 19044) and data type object
```

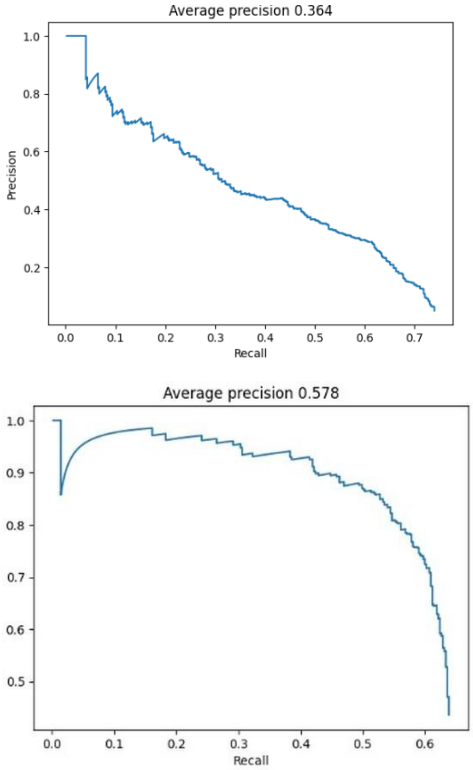
Unde primeam erori de la interpretor ca nu va reusi sa antreneze modelul pentru un numar extrem de mare de feature-uri

Sau, sa ajung fara RAM



Se intampla sa ajung si la 13000 MB
Din acel moment dadea crash, si anula tot progresul

Dar in final am reusit sa gasesc niste parametri care sa imi ofere precizii mai bune



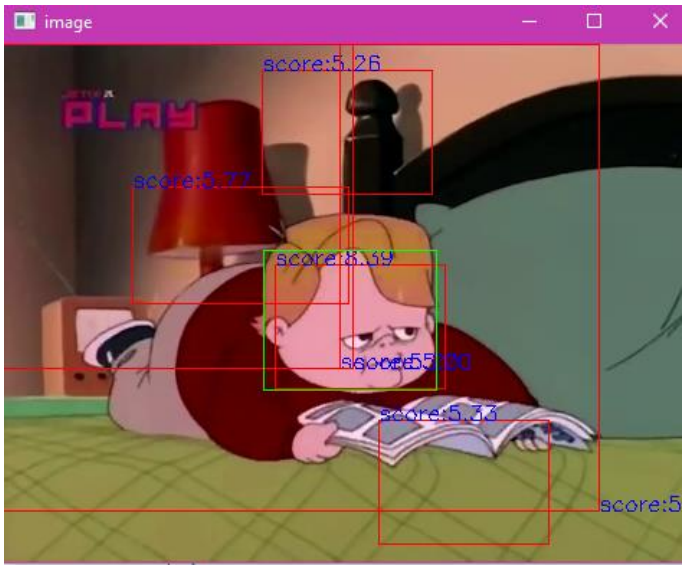
Acestia fiind parametrii pe care am reusit sa ii obtin, cu ajutorul carora detectam bine majoritatea fetelor dreptunghiulare

```
# parameters

dim_hog_cell = 9
dim_window = 51
overlap = 0.3
number_positive_examples = 7236
number_negative_examples = 15000
has_annotations = False
threshold = -1
cells_block = (2,2)
orientations=20

pixels_cell = (dim_hog_cell,dim_hog_cell)
resize_factors = [2,1.8]

resize_param = (int(dim_window*resize_factors[0]),int(dim_window*resize_factors[1]))
```

Dar inca nu era destul de bine pentru a considera proiectul finalizat

Prin urmare din acest moment am decis sa ma ocup de multiscaling, partea care din perspectiva mea, a durat mult mai mult decat m-am asteptat.

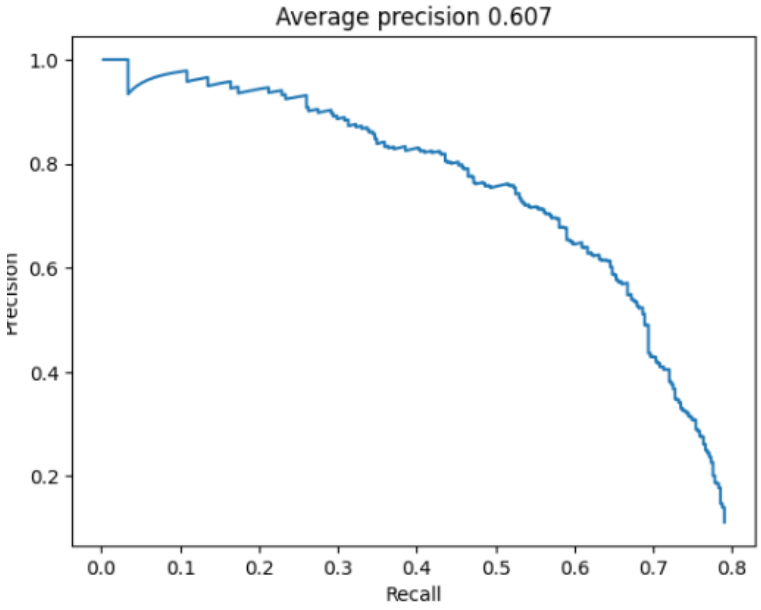
Primele incercari de multiscaling:



La finalul implementarii multiscaling-ului / piramidei de imagini, algoritmul rula pe imaginile redimensionate cu urmatorii factori.

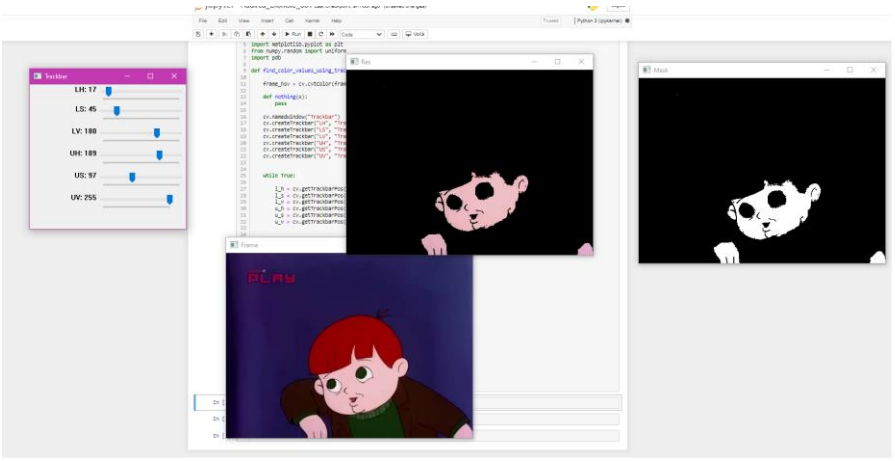
```
for factor in [0.25,0.33,0.5,0.75,1,1.25,1.5,2]:
    img = cv.resize(test_img,(0,0),fx=factor,fy=factor)
```

Astfel am reusit sa obtin primul scor > 0.60

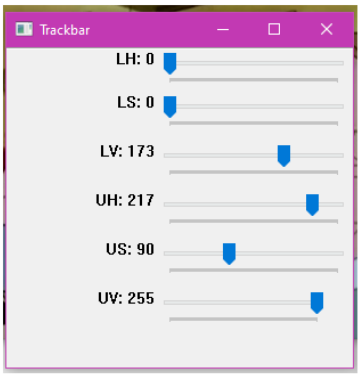


In timpul implementarii multiscaling-ului mi-a venit ideea de a utiliza in algoritm detectia cu ajutorul culorilor, astfel mi-am dat seama ca majoritatea personajelor din desen au culoarea pielii in medie roz. Astfel am decis sa filtrez culorile din imagine cu ajutorul reprezentarii lor in format HSV.

Am decis sa iau niste imagini aleator din setul de antrenare si sa aleg valorile potrivite pentru crearea mastii propriuzise, in care obtin 0 in locurile in care nu se afla culoare roz, si 1 in locurile in care exista nuante de culoare roz



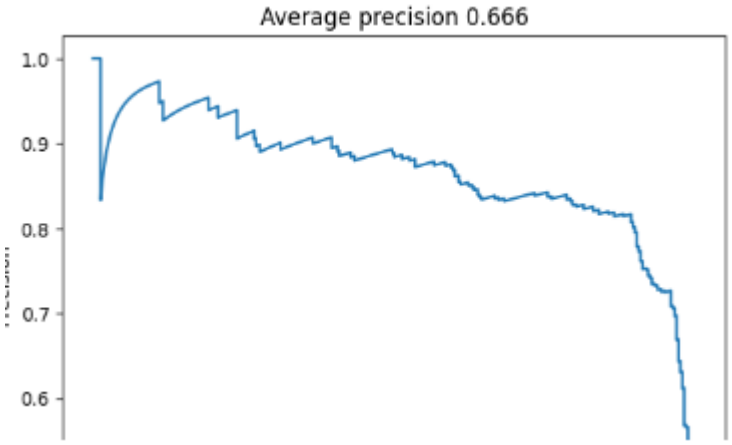
In final am ajuns sa obtin urmatoarele valori:



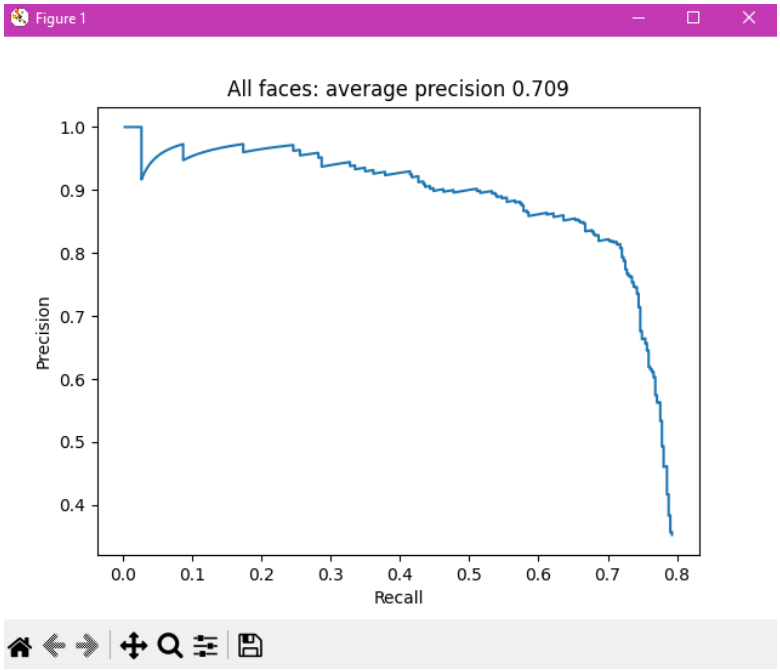
+ 20 la US pentru ca era prea putin pentru unele imagini

Iar in cod, patch-urile care au o valoare < 70 sunt omise de detectie

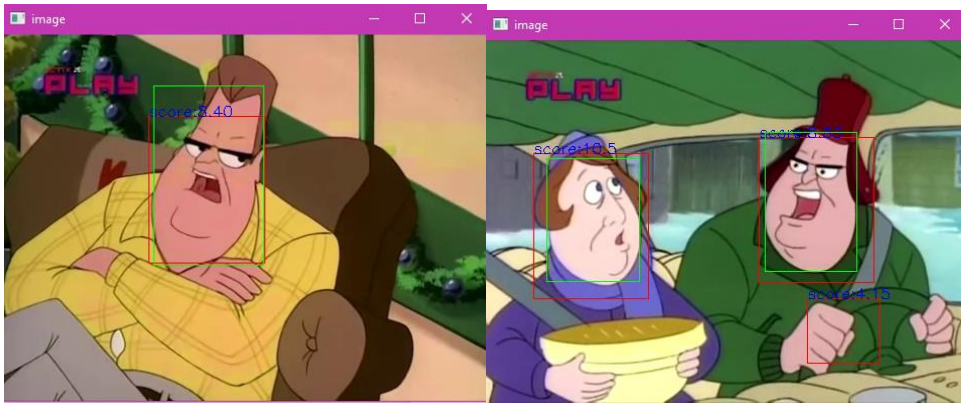
Nu puteam sa nu includ acest scor in documentatie



Dar stiam ca se poate scoate un scor mai bun, iar dupa o alta serie de trial and error cu diversi parametri am ajuns la scorul final 0.709

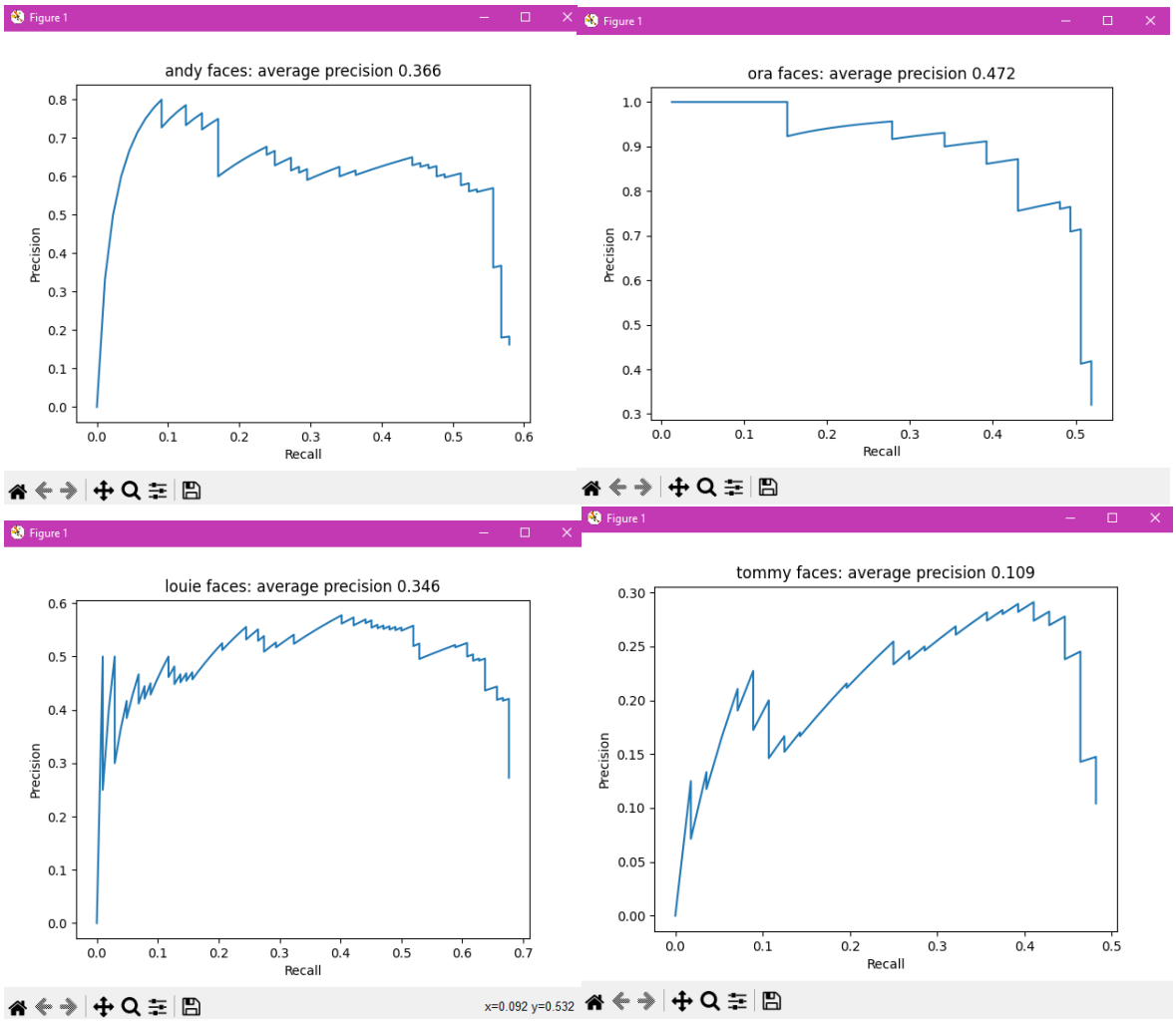


Voi plasa cateva imagini din detectiile care au obtinut scorul final.



Cu siguranta aceasta parte poate fi imbunatatita, si va fi imbunatatita in viitor

Facial Recognition



Pentru facial recognition cum am mentionat mai sus in documentatie, era necesar sa recunoastem fetele din imagine: Andy, Louie, Tommy, Ora

Caracteristicile au fost extrase folosind urmatorii parametri

```
classifier = LinearSVC(C = 10, max_iter=100000)


dim_hog_cell = 12
dim_window = 144
overlap = 0.3
number_positive_examples = 7236
number_negative_examples = 20000
has_annotations = False
threshold = 0
cells_block = (4,4)
orientations=20
```

Si am rulat modelul pe fetele detectate de algoritmul de face detection

Rezultatele pot fi observate mai sus

Cu siguranta si aceasta parte poate fi imbunatatita, si va fi imbunatatita in viitor