Laboratorul lui Dexter

Daria Pîrvulescu

Ianurie 2024

1 Scopul proiectului

Proiectul presupune rezolvarea a doua cerințe de recunoaștere și detecție facială cu ajutorul procedeului $sliding - window$ plus un task bonus:

- **Task1:** Detectarea facială a tuturor personajelor din desenul animat *Laboratorul lui Dexter* (prag acuratețe: 0.80%);
- **Task2:** Recunoașterea facială a patru persoanje din desenul animat *Laboratorul lui Dexter*: **Dad, Deedee, Dexter si Mom** (prag acuratețe: 0.60%);
- **Task Bonus:** Folosirea unui detector $state - of - the - art$ care să depașească pragurile menționate mai sus.

2 Analiza Datelor

Prima și cea mai importantă etapă în începerea unui astfel de proiect este analiza datelor. La o simplă trecere prin imaginile de train se poate observa destul de bine diferența între aspectul fețelor personajelor. Fiind un desen animat, fețele sunt exagerate și caricaturizate, astfel am putut extrage informația că o singură fereastră pătrată nu o să încadreze bine fețele tuturor. Acest lucru se poate vedea și în figura de mai jos Figura 1:

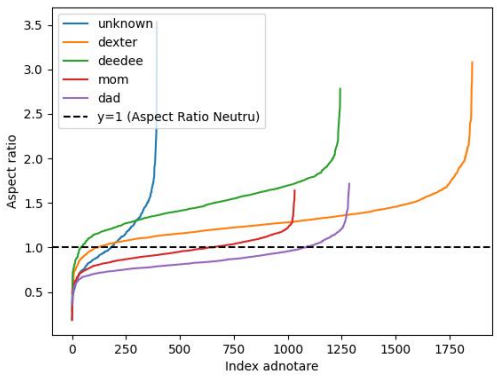

Figure 1

Un aspect ratio pedominant sub 1 înseamnă o fața lungă (Dad și Mom), iar unul predominant peste 1 înseamnă o fața lată (Deedee). Dupa cum se vede și în Figura 1 liniile nu sunt bine

delimitate de $y = 1$, ceea ce înseamnă că pentru fiecare persoanj trebuie ca $sliding - window$ -ul să aibă trei dimensiuni diferite: **pătrat, dreptunghi înalt și dreptunghi lat** pentru a maximiza detecțiile faciale.

3 Ideea de rezolvare

Pentru acest proiect am ales să folosesc modelul SVC pe care l-am antrenat folosind $descriptori - pozitivi$ si $descriptori - negativi$. Am antrenat câte un SVC pentru fiecare formă de fereastra, dar și pentru fiecare personaj în parte. Astfel am 3 SVC-uri pentru $task1$ și 12 SVC-uri pentru $task2$. După antrenarea modelelor am folosit procedeu $sliding - window$ pentru a detecta fețe, respectiv pentru a recunoaște fețe. Mărimea ferestrei pornește aproximativ de la un $zoom - in$ minimal de 1.2 , facând un $zoom - out$ până când ferestrele nu mai încap în imaginea micșorată. Am testat doua modele SVC: $liniar$ și cu kernel rbf.

4 Prelucrarea datelor

Notebook-ul $extrage - patch - poz$ se ocupă cu prelucrarea adnotărilor pentru a crea un fișier cu fețele decupate din datele de train. Am observat că adnotările sunt destul de fixate pe fața personajelor, de aceea pentru a putea generaliza mai bine am decupat fețele cu 10 % mai mult din imagine (acolo unde adnotarea îmi permitea sa nu ies din poză).

```
margine = 0.1
latime_extra = (x_max - x_min) * margine
inaltime_extra = (y_max - y_min) * margine


new_x_min = max(0, int(x_min - latime_extra))
new_y_min = max(0, int(y_min - inaltime_extra))
new_x_max = min(width, int(x_max + latime_extra))
new_y_max = min(height, int(y_max + inaltime_extra))


# Extrage patch-ul
patch = img[new_y_min:new_y_max, new_x_min:new_x_max]
```

Cum am descris și mai sus, fiecare fața are un $aspect - ratio$ ($\frac{latime}{inaltime}$) diferit în funcție de personaj, dar poate varia și îm intermediu aceluiași personaj. Toate adnotarile urmează să fie redimensionate la anumite mărimi ce diferă de marimile lor inițiale, deformând imaginile. Deformarea imaginilor poate conduce la o antrenare dezavantajoasă pentru SVC, de aceea am ales să le grupez dupa $aspect - ration$ în imagini **pătrat, dreptunghi-înalt** și **dreptunghi-lat**. Marimile sunt destul de permisivie astfel încât doar imaginile care urmează să fie semnificativ deformate să fie eliminate.

```
min_patrat,max_patrat= 0.7, 1.4
min_dreptunghi,max_dreptunghi= 1, 3.5
min_dreptunghi_inalt, max_dreptunghi_inalt = 0.5, 1.1
```

5 Descriptorii

5.1 Descriptorii pozitivi

După pregătirea datelor, am făcut descriptori pozitivi separați pentru fiecare personaj ($+unknown$) și pentru fiecare formă de fereastră. Astfel, pentru $task1$ i-am concatenat pe toți, iar pentru $task2$

i-am folosit separt pentru recunoașterea personajelor. Pentru *task*1, descriptorii pozitivi de la *unknown* sunt folosiți ca date pozitive, iar pentru *task*2 îi concatenez cu descriptorii negativi pentru a-i folosi pe post de date negative.

Pentru a mări numărul de exemple pozitive, pe lângă adnotările extrase din datele de antrenare, am pus și adnotările întoarse pe axa verticală și adnotările rotite cu 7 grade.

5.2 Descriptorii negativi

Am creat câte un descriptor negativ pentru fiecare forma de fereastră. Descriptorii negativi sunt luați random din imaginile de antrenare și au mărimi variabile random între 20 și 270 de pixeli. Pentru a evita situația în care, generând random *pach*-uri acestea vor cădea fix pe fața unui personaj, adaugând o imagine pozitivă la cele negative, am verificat înainte de adăugare daca *Intersection Over Union* (IOU) este mai mică de 0.3. De asemenea, pentru a micșora entropia, am ales să compar un nou *patch* ales cu celelate alese până la acel moment. Dacă IOU-ul este mai mare de 0.3, trec mai departe.

Un efect negativ al generării aleatoare de imagini este faptul ca acestea se pot repeta sau conține informații nesemnificative pentru modelul antrenat. De aceea pentru a crește probabilitatea găsirii unor descriptori negativi "buni", am ales să generez un număr destul de mare de astfel de imagini, undeva între 30.000 și 50.000.

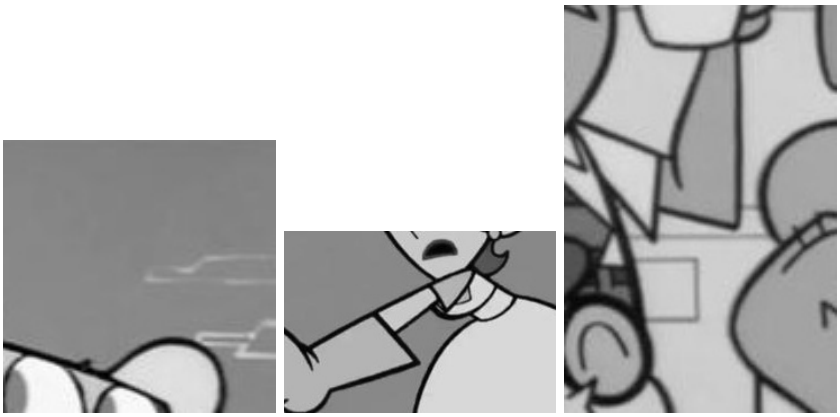

Figure 2: Exemple de imagini folosite la formarea de descriptori negativi

5.3 Hog

Am ales să experimentez cu mai multe configurații de funcții *hog* pentru a vedea ce se prestează cel mai bine pentru setul meu de date. Combinația care mi-a dat rezultatele cele mai bune este urmatoarea:

```
descriptor = hog(patch, pixels_per_cell=(12,12),
                 cells_per_block=(2, 2), feature_vector=True,
                 block_norm='L2-Hys',orientations = 12 )
```

Am ales ca $pixels-per-cell$ să fie 12 pentru ca informația găsită de SVC să fie mai generală, iar *orientations* de 12 pentru a împărți intervalul unghiurilor gradientului în 12 *bin*-uri și pentru a capta detaliile fine din fetele personajelor. Dacă ne întoarcem la analiza datelor, alegerea acestor valori ale lui *hog* poate fi justificată de faptul că personajele au anumite particularități comune pe care modelul de $machine-learning$ le poate ușor confunda. Spre exemplu: Dad si Dexter au aceleași forme de ochelari, Deedee are ochii destul de mari și rotunzi în forma ochelarilor

personajelor anterioare. Am ales, totuși ca $pixels - per - cell$ să fie 12 pentru a generaliza puțin informația. Personajele sunt caricaturizate și pot exista diferențe semnificative de la o imagine la alta la același persoanj (ex: "evil Dexter" care are ochii în formă de triunghi vs "normal Dexter").

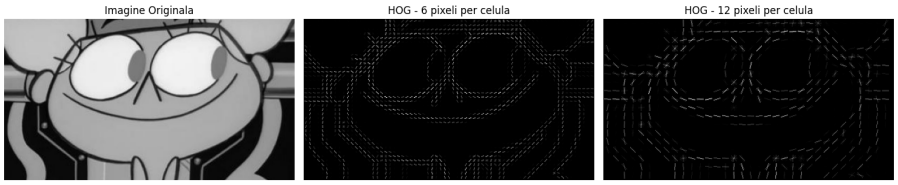

Figure 3: Vizualizarea imaginilor în funcție de mărimea celulei hog

6 SVC

Am testat două tipuri de SVC-uri: *liniar* și kernel rbf. Codul pentru ambele este aproape identic și se găsește în notebook-ul *antrenare*. Trec prin mai multe valori de C, dar de obicei $C = 1$ rezultă în cel mai bun clasificator. Pentru **task1** antrenez 3 clasificatoare, unul pentru descriptorii patrat, unul pentru cei dreptunghi lat, respectiv unul pentru cei dreptunghi înalt. Descriptorii negativi sunt cei calculați anterior pentru fiecare formă geometrică și cei pozitivi sunt concatenarea tuturor descriptorilor pozitivi per personaj plus *unkown*.

```
descriptori_poz = np.concatenate([descriptori_poz_dad,descriptori_poz_deedee,
    descriptori_poz_dexter,descriptori_poz_mom,descriptori_poz_unknown])
```

Pentru **task2** antreneaz 12 clasificatori, 3 (pătrat, dreptunghi lat, dreptunghi înalt) pentru fiecare dintre cele 4 personaje. Descriptorii pozitivi pentru personaje sunt cei descrisi mai sus la sectiunea Descriptori pozitivi, iar descriptorii negativi sunt compuși din descriptorii negativi formei aferente, descriptorii pozitivi pentru personajele *unknown* și **descriptorii pozitivi ale personajelor încurcate frecvent cu personajul SVC ului antrenat**

```
#descriptor negativ dexter
descriptori_neg_anti = np.concatenate([descriptori_poz_dad,descriptori_poz_mom,
    descriptori_poz_unknown,descriptori_neg])
```

Am încercat să pun doar personajele confundate destul de mult cu personajul căutat, deoarece nu vreau ca modelul să "învețe" că fețele sunt exemple negative în general. Vreau să "învețe", spre exemplu, că fețele Mom și Dad nu sunt $Dexter$.

7 Sliding-window

Baza proiectului a fost folosirea procedeului de $sliding - window$. Pentru fiecare imagine, cât timp ferestrele încap în imagine, folosesc cele 3 tipuri de SVC-uri pentru a face predictii. Detectiile care depășesc un anumit prag pozitiv al scorului (un scor negativ înseamnă o "non-față") și scorurile sunt adunate de la fiecare SVC într-un singur $np.array$, urmând să aplic **Non Maximal Suppression**(NMS) pe ele. NMS-ul elimină detecțiile care se suprapun prea mult ($IOU \geq 0.3$) și definește predicția cea mai precisă. Această logică se aplică atât pentru $task1$ cât și pentru $task2$ (pe SVC-urile aferente), deoarece tratez task-urile independent și încerc să

”găsesc” personajele la fel cum am ”găsit” și fețele, în contrast cu detecția facială urmată de decizia a cui personaj îi aparține.

Mai jos sunt câteva grafice de acuratețe obținute cu diferite valori de ferestre, dimensiune de celula hog și SVC-uri.

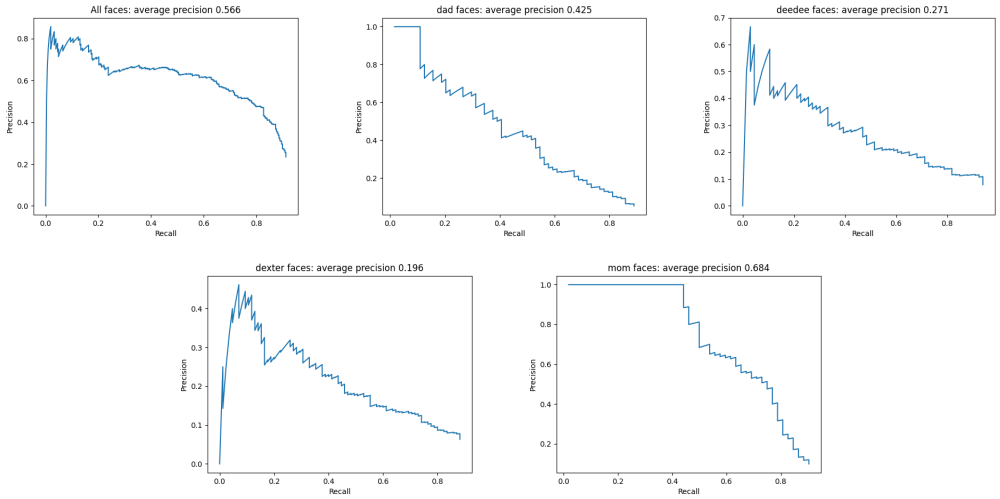

Figure 4: SVC-liniar, dim patrat = 96x64, dim dreptunghi lat = 96x64, dim dreptunghi lung = 64x96, dim hog cell = 12, orientare = 12

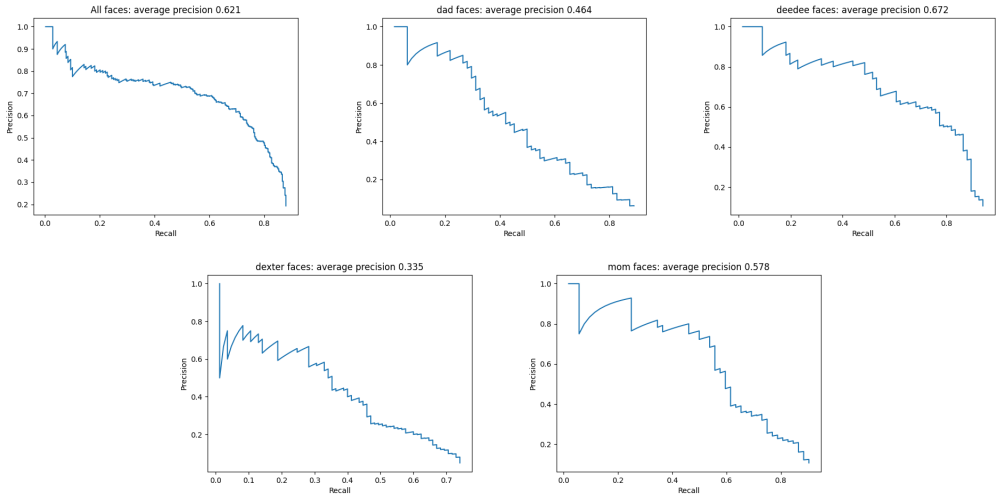

Figure 5: SVC-liniar, dim patrat = 128x128, dim dreptunghi lat = 128x86, dim dreptunghi lung = 86x128, dim hog cell = 8, orientare = 9

Printre alte valori încercate pentru SVC ul liniar se află și fereste de $48x64$ și $36x54$ cu dimensiunea celulei hog de 6, 8, 10 și 12 și cu număr de descriptori ce varia între 20.000 și 60.000, însă scorurile au fost mici, netrecând de un prag de 50%.

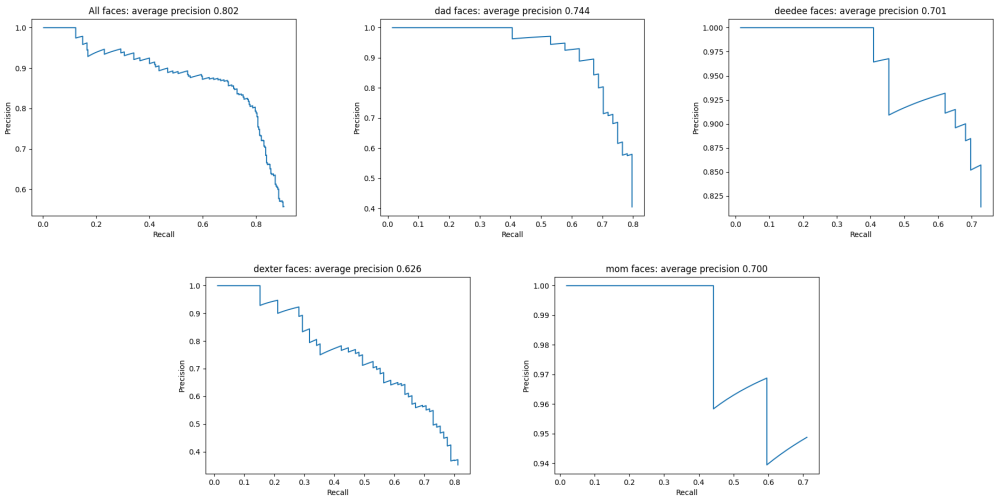

Figure 6: Varianta finală: SVC-kernel-rbf, dim patrat = 96x96, dim dreptunghi lat = 96x64, dim dreptunghi lung = 64x96, dim hog cell = 12, orientare = 12

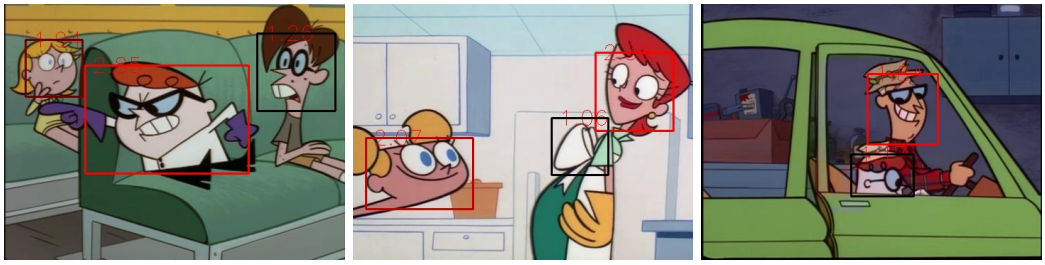

Figure 7: Diferite detecții realizate cu varianta finală a proiectului

8 Bonus-YOLO

YOLO ($You - only - look - once$) este un clasificator $state - of - the - art$ bazat pe *Rețele Neuronale Convolutionare*, având 24 *convolutional layers*, 4 *max-pooling layers* si 2 *fully connected layers*. YOLO tratează întreaga imagine ca un *input* și generează direct predicții pentru bounding box-uri și clase, folosind o rețea CNN, procesând întreaga imagine dintr-o singură trecere. Astfel, acesta poate fi folist cu o acuratețe ridicată pentru *real time image detection*.

Fiecare imagine este împărțită într-o grilă de dimensiuni SxS, iar fiecare celulă a grilei este responsabilă de prezicerea unui număr fix de bounding box-uri, împreună cu scorurile de încredere și probabilitățile pentru clasele detectate. Acest proces reduce complexitatea și asigură o integrare fluidă între localizarea și clasificarea obiectelor.

Pregatirea datelor este un punct important în folosirea modelului, deoarece $YOLO$ are un mod specific prin care primeste datele. Folderul *dataset* este alcatuit din:

- train
  - images
    - 0001.jpg
    - ...

  - labels
    * 0001.txt
    * ...

- val
  - images
    * 0001.jpg
    * ...
  - labels
    * 0001.txt
    * ...

Dupa cum se vede mai sus, fiecare imagine are un *label* propriu în care apar pe rând *clasa* obiectului, *x-center si y-center* coordonatele centrului adnotării noramlizate și *width, height* lățimea și lungimea adnotării normalizate.

```
3 0.12604166666666666 0.30694444444444446 0.20625 0.24166666666666667
0 0.621875 0.30833333333333335 0.21875 0.36666666666666664
2 0.428125 0.725 0.16458333333333333 0.17222222222222222
```

Pe lângă *dataset*, modelul mai primește și un *date.yaml* pe care îl folosește să descifreze clasele obiectelor și *path*-ul către *dataset*.

Modelul a fost antrenat pe *kaggle* (pentru a beneficia de *GPU*).

Figure 8: Antrenarea modelului YOLO

Am antrenat 2 modele separte de *YOLO*, unul pentru *task*1 respectiv unul pentru *task*2, singura diferență între ele este număr de clase pe care sunt antrenate: una (față), respectiv patru (Dad, Mom, Dexter și Deedee).

Dupa antrenare, predicțiile sunt salvate în variabila *results* și distribuite către *detectii*, *scoruri* și $file - name$, unde *model* reprezintă *weights.best.pt*

```
model = YOLO("./YOLO_personaje/weights/best.pt")
...
results = model.predict(source=img, save=False, verbose=False)
```

Rezultatele sunt unele extrem de bune:

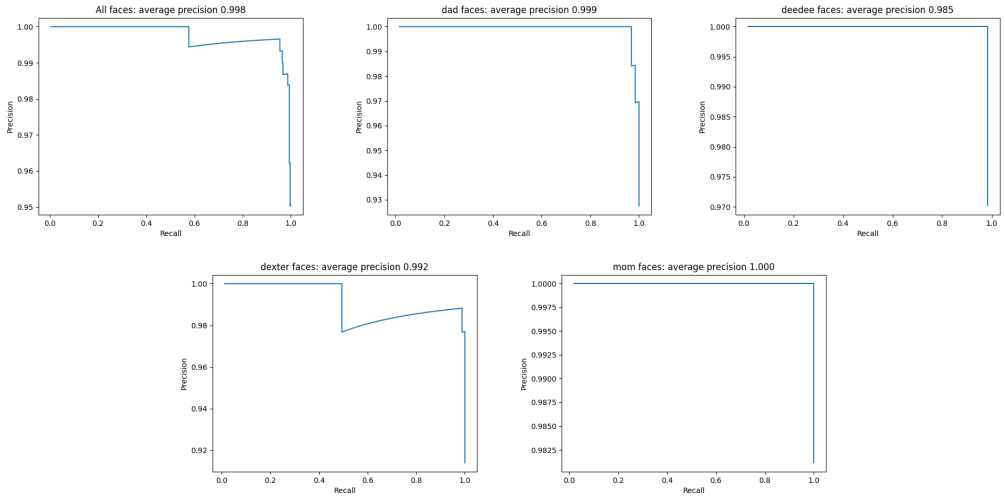

Figure 9: Varianta finală: YOLO yolov8n.pt, epoci = 50, batch = 16

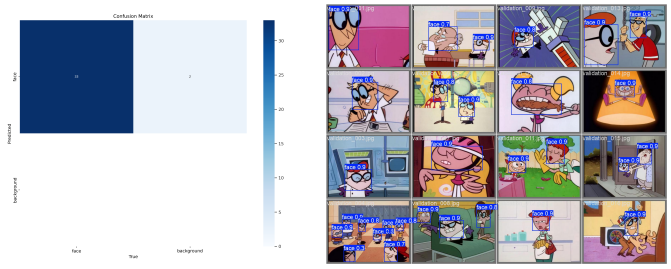

Figure 10: Alte metrici extrase pentru $task1$

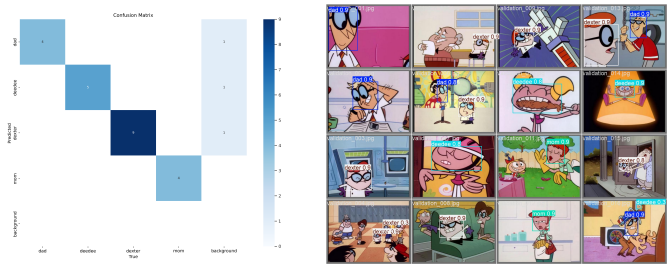

Figure 11: Alte metrici extrase pentru $task2$