

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

*Disciplina Concepte și Aplicații în Vedere Artificială*

Universitatea din București  
Facultatea de Matematică și Informatică  
Iulia-Georgiana Talpalariu  
Grupa 334

Ianuarie 2024

## 1 Introducere

Spre rezolvarea taskurilor de vedere artificială de la tema *Detectarea și recunoașterea facială a personajelor din serialul de desene animate Familia Flintstone*, propun o soluție documentată și detaliată în raportul de față. Soluția mea utilizează noțiunile dobândite la disciplina *Concepte și Aplicații în Vedere Artificială* și este implementată în limbajul Python cu ajutorul librăriilor numpy, opencv, pytorch, skimage, sklearn.

## 2 Prezentarea datelor

Datele de antrenare reprezintă 4000 de imagini de dimensiune **480 x 360**, color, 1000 în care sigur se află cel puțin unul din personajele **Barney**, **Betty**, **Fred** și **Wilma**, eventual alături de alte personaje, extrase din episoade din serialul **Familia Flintstone**. Datele de validare se prezintă sub forma a 200 de imagini dimensiune **480 x 360**, color.



Figura 1: Imagine de antrenare

Alături de aceste imagini există fișiere de **adnotări**. Pentru fiecare din cele 4 personaje, există câte un fișier de adnotări care conține pe fiecare linie (Fig.2):

- numele fișierului
- 4 coordonate care delimită deosebită detecția unei fețe reprezentând  $x_{min}, y_{min}, x_{max}, y_{max}$
- numele personajului dacă acesta face parte din cele 4 mai sus amintite, altfel eticheta *unknown*

```
0302.jpg 79 155 144 211 barney
0302.jpg 173 106 235 179 fred
0302.jpg 320 102 382 171 unknown
```

Figura 2: Adnotarea corespunzătoare imaginii din figura 1

### 3 Prezentarea taskurilor

Proiectul cuprinde rezolvarea celor 2 taskuri:

**Task 1 Detecția facială** a personajelor din Familia Flinstone prin metoda ferestrei glisante (*sliding window*).

**Task 2 Recunoașterea facială** a personajelor **Barney, Betty, Fred și Wilma**.

### 4 Metrici de evaluare

Performanța este evaluată la taskul 1 cu ajutorul ariei de sub curba **precizie-recall** (precizie medie sau *average precision = AP* ), iar la taskul 2, cu media ariilor de sub curba **precizie-recall** pentru cele 4 personaje (*mean average precision = mAP*).

### 5 Metode abordate

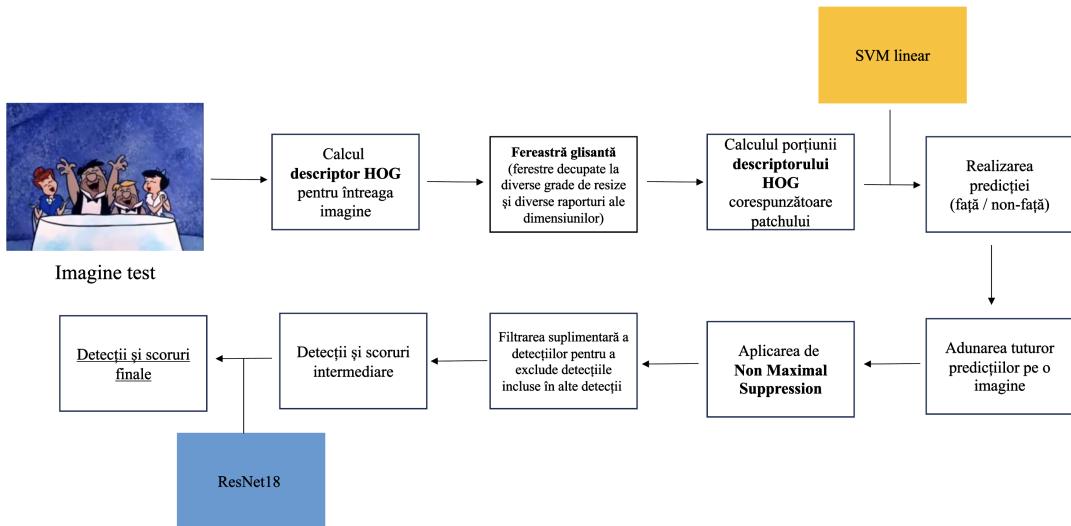


Figura 3: Viziune de ansamblu asupra soluției oferite de mine

## 5.1 Modele utilizate

Pentru rezolvarea taskurilor 1 și 2, am utilizat:

- **SVM (Support Vector Machine)** ( `sklearn.svm.LinearSVC` )
- **ResNet18** ( `torchvision.models.resnet18` )

**La taskul 1**, folosesc o **cascadă de clasificatori** astfel:

- SVM antrenat pe descriptori HOG
- ResNet18 antrenat pe detectiile decupate (color) din imaginile de antrenare

**La taskul 2**, folosesc 4 rețele ResNet18 câte una pentru fiecare dintre personajele **Barney**, **Betty**, **Fred** și **Wilma** - care clasifică binar dacă respectiva detectie reprezintă sau nu fața aceluia personaj, bazându-mă pe detectiile intermediare de la taskul 1.

## 5.2 Selectare exemple pozitive și negative

### 5.2.1 Task 1

Pentru antrenarea **SVM**, am folosit datele de antrenare date.

Pentru **exemplele pozitive**, am folosit adnotările pentru a decupa imagini cu fețele personajelor din toate imaginile de antrenare.

Pentru **exemplele negative**, folosesc 2 strategii: decupare automată, random de porțiuni din imaginile de antrenare (cu excluderea fețelor) și decupare manuală de porțiuni din imagini, după primele serii de antrenare și validare, porțiuni care includeau zone predispuze a fi confundate drept fețe (obiecte rotunde, mâini, picioare, brațe, îmbrăcăminte etc.).

Pe toate aceste exemple (pozitive și negative) le preprocesez pentru antrenare, precalculând **descriptorii lor HOG**. Exemplele pozitive sunt luate mereu toate și fiind fețe le-am putut augmenta aplicând un flip horizontal. Exemplele negative sunt amestecate (*shuffled*) înainte de a fi trimise la antrenare și ales un număr de exemple negative =  $1.2 * \text{numărul de exemple pozitive}$ .

Pentru antrenarea **ResNet18** la taskul 1 folosesc toate exemplele pozitive (detectiile decupate din imaginile de antrenare) color și toate exemplele negative adunate prin metodele menționate anterior (color).

### 5.2.2 Task 2

Pentru taskul al doilea, la antrenarea fiecărei rețele **ResNet18** din cele 4 specifice pentru fiecare personaj, folosesc:

**Exemple Pozitive** Toate imaginile cu detectiile de fețe ale personajului X din imaginile de antrenare

**Exemple Negative** De **două ori** mai multe decât exemplele pozitive:

- $\frac{1}{3}$  din imaginile cu celelalte personaje în afară de X
- Restul până la completarea numărului de exemple negative - exemple negative din cele alese automat random sau manual pentru taskul 1

## 5.3 Augmentarea datelor

Pentru a crește numărul exemplelor pozitive pentru antrenarea SVM-ului am adăugat pe lângă detectiile decupate din setul de antrenare și oglinditele lor stânga-dreapta.

Pentru a avea diversitate mai mare, la antrenarea ResNet18 (taskurile 1 și 2), aplic o transformare care oglindește tot stânga-dreapta o imagine cu probabilitate de 50% :

```
torchvision.transforms.RandomHorizontalFlip(p=0.5)
```

## 6 Soluție

Am realizat o soluție în limbajul Python, folosind diverse librării. Pentru antrenarea **ResNet18** am folosit un notebook pe platforma **kaggle**. Pentru taskul 1, am realizat o clasă, **FacialDetector**, iar pentru al doilea task, o altă clasă, **FacialRecognizer**, care se pot găsi în codul atașat, în fișierele omonime.

### 6.1 Soluția pentru taskul 1

Pentru taskul 1, am început prin a adapta metoda folosită la laborator, care utilizează metoda ferestrelor glisante pentru a rezolva taskul de detecție via clasificare. Metoda utilizată la laborator era bazată pe calcularea descriptorilor HOG pentru fiecare imagine, parcursa imaginilor folosind metoda ferestrelor glisante și clasificarea fiecărei ferestre folosind un clasificator liniar.

#### 6.1.1 Descriptorii HOG

Atât în etapa de antrenare, cât și în cea de testare trebuie să calculăm descriptorii HOG (*Histogram of Oriented Gradients*). La antrenare, descriptorii HOG sunt calculați pe fiecare imagine care conține sau nu o față (ocupată din datele de antrenare). La testare calculez descriptorul HOG pentru întreaga imagine de test și parcurg celulele HOG cu ferestre de dimensiune fixă. O parte considerabilă de timp dedicată rezolvării proiectului a fost investită în alegerea parametrilor pentru descriptorul HOG. Cea mai bună performanță obținută exclusiv cu HOG, fereastră glisantă și clasificator liniar (SVM liniar) a fost folosind parametrii enumerați în tabela 1. Am obținut  $AP = 72.3\%$ .

Totuși, când am utilizat ResNet18 pentru taskul 1, folosindu-l ca să clasifică două oară detecțiile obținute în mod clasic cu SVM și HOG, am obținut un scor mai mare final când am folosit dimensiunea ferestrelor de 48 X 48 pixeli, restul parametrilor păstrându-i constanți și obținând un descriptor de dimensiune **2352**.

Parametru	Valoare
Dimensiune fereastră	54 x 54 pixeli
Dimensiune celulă hog	6 x 6 pixeli
Număr orientări	12
Normalizare per bloc	L2
Celule per bloc	2 x 2
Total dimensiune descriptor HOG	3072

Tabela 1: Parametrii pentru descriptorul HOG - scenariu pentru care cu SVM cu  $C = 0.1$  și pragul scorului de 3.5 obțin scorul de **72.3 %**

#### 6.1.2 Algoritmul de fereastră glisantă

Am îmbunătățit algoritmul de fereastră glisantă folosit la laborator, modificând imaginea de test prin aplicarea de redimensionări și transformări ale raportului  $\frac{h}{w}$  (*aspect ratio*,  $h = height$  = înălțime,  $w = width$  = lățime). Pentru redimensionări ale întregii imagini de test am folosit 35 de *scale*:

```
resize_scales = [x / 100 for x in range(5, 30, 3)] + [x / 100 for x in range(20, 150, 5)]
```

Pentru schimbarea raportului  $\frac{h}{w}$  am utilizat următoarele valori:

```
aspect_ratios = [0.6, 0.8, 0.9, 1, 1.1, 1.2, 1.3]
```

Astfel, imaginea de test se modifică, iar noi parcurem cu aceeași dimensiune de fereastră, simulând modificarea (*resize* și *aspect ratio* diferit) pe fereastră.

Puteam obține transformarea din coordonate în imaginea modificată în coordonate în imaginea inițială astfel:

```
x_min = int((x // w_scale) * self.params.dim_hog_cell)
y_min = int((y // h_scale) * self.params.dim_hog_cell)
x_max = int((x * self.params.dim_hog_cell + self.params.window) // w_scale)
y_max = int((y * self.params.dim_hog_cell + self.params.window) // h_scale)
```

### 6.1.3 Antrenare SVM și clasificare

Pentru clasificator, am ales **LinearSVC**, obținând cele mai bune performanțe când am antrenat cu **C=0.1**, C reprezentând gradul în care penalizăm eroarea, astfel că un C de valoare mai mare (1, 10, etc.) reușește să clasifice (aproape) perfect exemplele de antrenare, dar obține rezultate mai slabe pe exemplele de test (*overfitting*).

Clasificarea se realizează calculând scorul dat de SVM, antrenat în modul descris anterior, pe descripțorii aferenți fiecărei ferestre din imaginea de test:

```
descriptor_window = hog_descriptors[y:y + num_cell_in_template,
                                     x:x + num_cell_in_template].flatten()
```

```
score = np.dot(descriptor_window, w)[0] + bias
```

unde  $w$  sunt ponderile clasificatorului LinearSVC. Detectia este considerată **față** dacă scorul trece de **pragul de 3.5** pentru situația în care se utilizează doar SVM și **pragul de 3**, atunci când folosim și filtrarea suplimentară folosind ResNet.

### 6.1.4 Filtrare

Deoarece clasificarea folosind HOG și SVM obține foarte multe detecții suprapuse și destul de multe detecții fals pozitive, folosesc mai multe metode de a filtra ferestrelle cu fețe obținute de SVM.

**Suprimarea non-maximelor** Suprimarea non-maximelor (*Non Maximal Suppression*) reprezintă o tehnică de filtrare a detecțiilor obținute, pentru a elimina detecțiile duplicate și a păstra detecțiile relevante (cu scoruri mai mari). Pentru această tehnică am utilizat funcția de la laborator.

**Filtrare pentru eliminarea detecțiilor incluse** Pentru a elimina detecțiilor incluse în alte detecții cu scor mai mare, am implementat funcția `filter_detections` în clasa `FacialDetector`.

**Filtrare folosind un al doilea clasificator** Înainte de acest pas, salvez detecțiile, scorurile intermedie și numele fișierelor în 3 fișiere numite:

- `detections_all_faces_before_resnet.npy`
- `file_names_all_faces_before_resnet.npy`
- `scores_all_faces_before_resnet.npy`

**Scorurile acestea intermedie sunt cele date de SVM.**

Pentru a reduce detecțiile fals pozitive, am decis să antrenez o rețea convinguțională neuronală, mai exact **ResNet18** din **torchvision**, folosind algoritmul de coborâre pe gradient (*Stochastic Gradient Descent*), funcția de pierdere *cross-entropy*, cu rata de învățare 0.001 și regularizarea L2 (*weight decay*) de 0.001. Antrenarea se face pe **batch-uri de 32 de imagini** și durează **30 de epoci** pentru ResNet-ul din taskul 1. Această rețea este antrenată pe detecțiile decupate din imaginile de antrenare, caracteristicile folosite fiind chiar pixelii. Detecțiile folosite pentru antrenare au fost redimensionate (`scaler`) și normalize (`normalize`) pentru a fi input adecvate pentru rețea ResNet.

```
scaler = transforms.Resize((224, 224))
normalize = transforms.Normalize(mean=[0.485, 0.456, 0.406],
                                 std=[0.229, 0.224, 0.225])
```

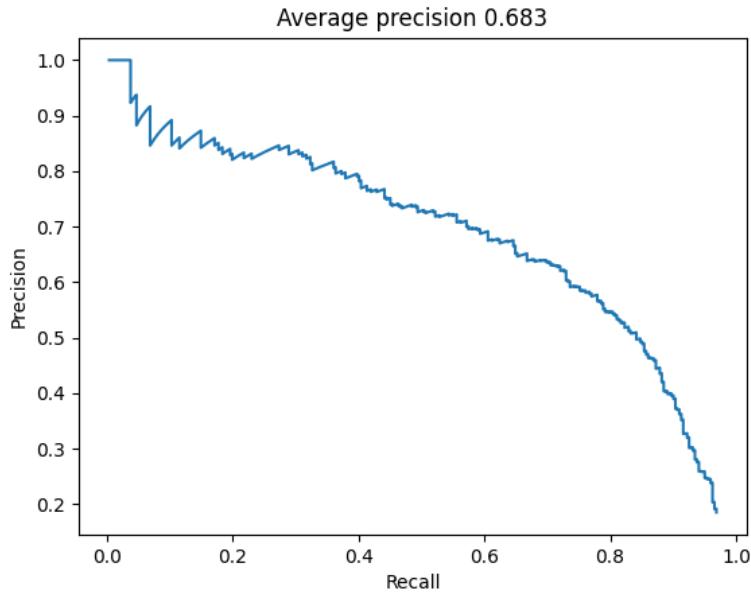
Scorurile finale sunt cele date de ResNet. Aria de sub curba precizie - recall este **92.8%** (Figura 4b).

**Avantajul** utilizării în cascadă a metodei de detecție facială cu SVM, apoi cu ResNet este în principiu timpul la testare, care este redus față de situația în care aș fi folosit direct rețea ResNet pe fiecare fereastră posibilă.

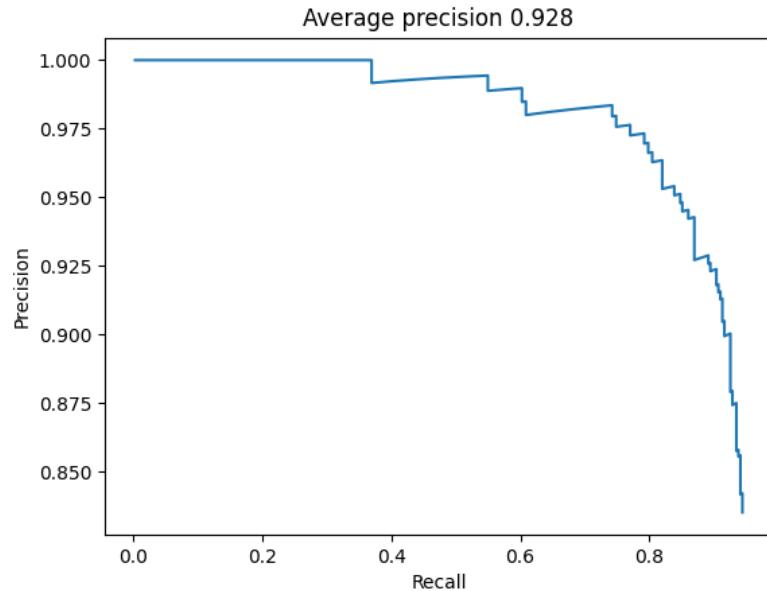
**Dezavantajele** sunt: antrenarea a două tipuri de clasificatori care necesită caracteristici diferite (descripтори HOG vs pixelii) și faptul că dacă o detecție nu a fost făcută cu SVM în etapa intermedie, ea nu va putea fi niciodată făcută ulterior cu ResNet, acesta evaluând și reclasificând doar detecțiile intermedie. De asemenea, am observat că dacă detecțiile din etapa intermedie sunt mai largi (cuprind nu numai față, ci și o parte din fundal sau corp), ResNet tinde să le dea acelora ferestre

scoruri mai mici sau chiar să le clasifice ca non-fete, deoarece el e antrenat exclusiv pe imagini cu fețe decupate extrase din setul de antrenare.

De asemenea, târziu mi-am dat seama că **nu am analizat distribuția fețelor** în seturile de antrenare și testare și, de aceea, nu am luat-o în considerare la realizarea soluției. Acest lucru era foarte important având în vedere că fețele cu Betty sunt subreprzentate față de celelalte și poate constitui un motiv pentru care ResNet-ul antrenat la taskul 1 le clasifică drept non-fete (6).

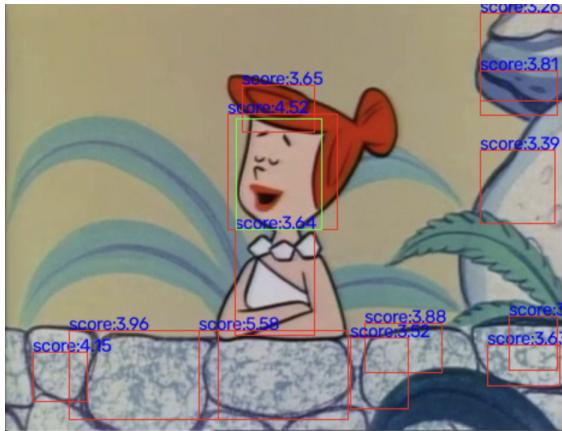


(a) Graficul curbei precizie - recall pentru varianta doar SVM (prag de 3), înainte de a aplica filtrarea cu ResNet

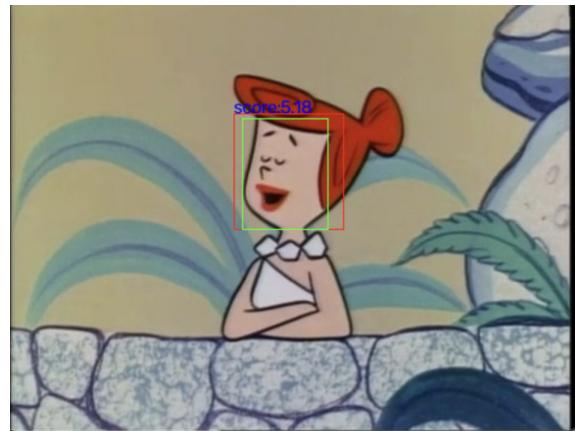


(b) Graficul curbei precizie - recall pentru varianta de SVM + ResNet

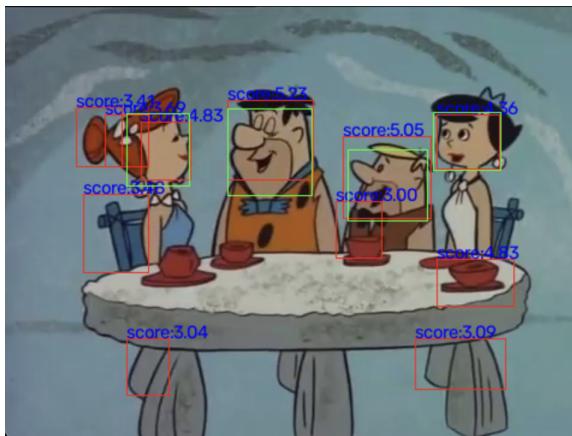
Figura 4: Graficul curbei precizie - recall la etapa intermediară (4a) și finală (4b)



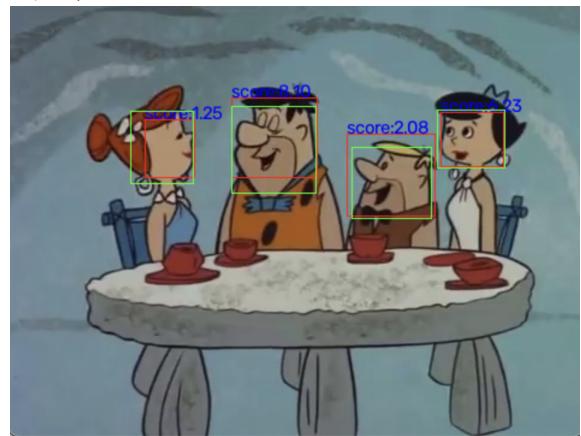
(a) Detectii in urma etapei intermediare - multe fals pozitive



(b) Detectie ramasa dupa filtrarea cu ResNet (clasificarea detectiilor intermediare cu ajutorul retelei)

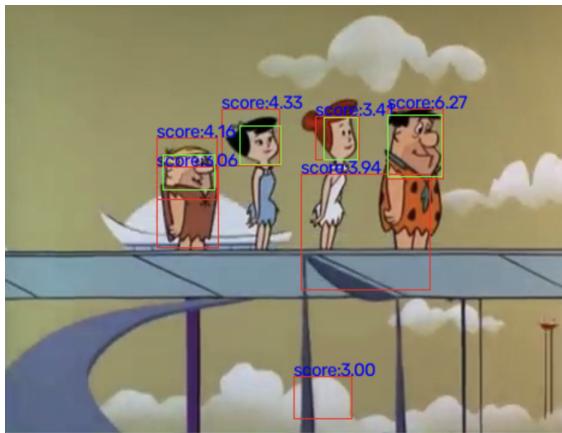


(c) Detectii in urma etapei intermediare - multe fals pozitive



(d) Detectie ramasa dupa filtrarea cu ResNet (clasificarea detectiilor intermediare cu ajutorul retelei)

Figura 5: Comparații între detectiile etapelor intermediare și finale



(a) Detectii in urma etapei intermediare - cateva fals pozitive, Betty e detectata, dar cu un chenar mai larg decat fata



(b) Detectie ramasa dupa filtrarea cu ResNet - Betty nu mai e detectata

Figura 6: Detectie pierduta din cauza filtrarii cu ResNet

## 6.2 Soluția pentru taskul 2

Pentru taskul al doilea, antrenez 4 rețele ResNet18, care clasifică binar fața personajului X / non-fața personajului X, rețelele configurate la fel ca rețeaua de la taskul 1, dar antrenate **50 epoci**. Pe detectiile și scorurile intermedii obținute la taskul 1 (folosind fereastră glisantă, HOG și SVM), pentru fiecare dintre cele 4 personaje (**Barney**, **Betty**, **Fred** și **Wilma**) rulez în mod de evaluare, rețeaua ResNet18 specifică fiecărui personaj pentru a obține detectiile pentru fețele aceluia personaj. Scorurile finale sunt cele date de ResNet și rezultatele sunt ilustrate în figura 7, obținând un *mean average precision* de

$$\frac{0.904 + 0.812 + 0.814 + 0.765}{4} = 82.375\%$$

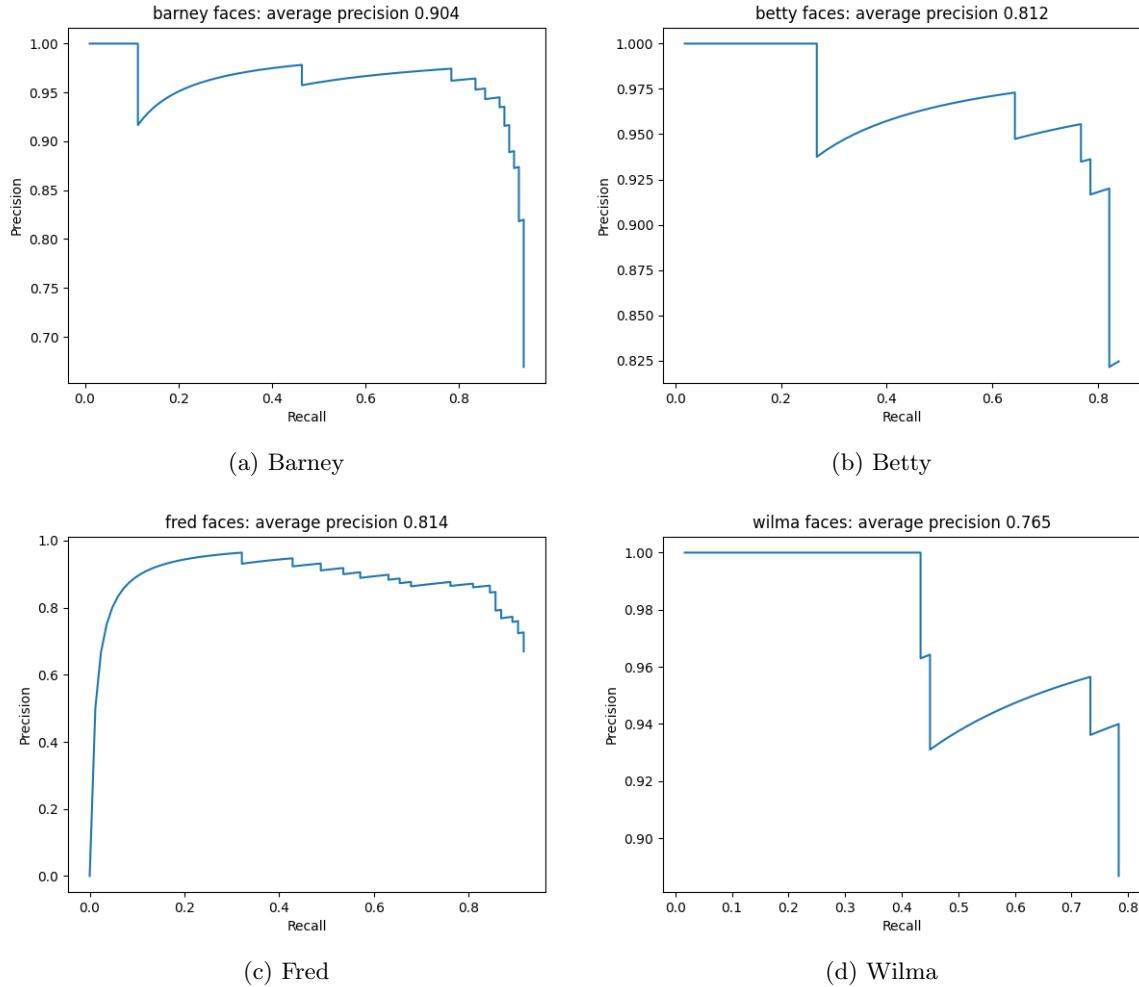
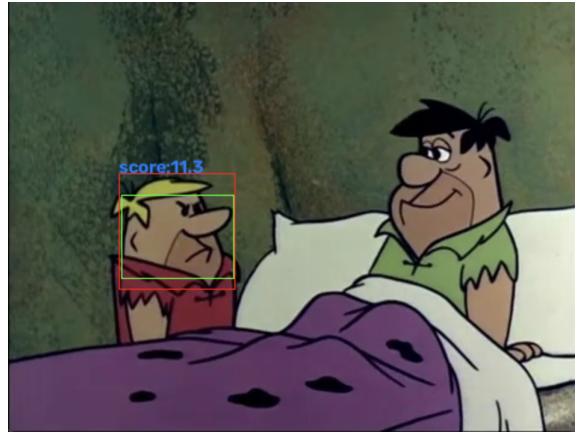


Figura 7: Graficul curbei precizie - recall pentru taskul 2 (Recunoaștere facială)



(a) Detectie multiplă personaj Barney



(b) Detectie corecta personaj Barney



(c) Barney detectie prea larga (include corpul)



(d) Betty detectie corecta (stanga) si confuzie cu Wilma (dreapta)

Figura 8: Diverse detectii pentru taskul de recunoastere faciala

## 7 Notă

În figurile 5, 6, 8 chenarele cu verde reprezintă detecțiile adevărate (*ground truth*) pentru imaginea dată și taskul respectiv (5 și 6 - detectie facială, 8 - recunoaștere facială), iar chenarele cu roșu, respectiv scorul asociat reprezintă detecțiile date de algoritmul meu (la etapa intermedieră scorul e cel dat de SVM, iar la etapa finală, scorul e cel dat de rețeaua ResNet).

## 8 Concluzii

Ambele taskuri, de detectie și de recunoaștere facială, sunt interesante din perspectiva faptului că există multiple abordări și se pot combina abordările în diverse moduri pentru îmbunătățirea rezultatului. Pentru proiectul de față, cele mai mari provocări au fost: de a găsi parametrii potriviti la calculul descriptorilor HOG pentru o detectie decentă (  $AP \geq 60\%$  ) și de a găsi metode de a reduce detecțiile fals-pozițive.