Flintstones face detector

Stan Cătălin-Andrei

Cuprins

1 Task 1

1.1 Modelul folosit

Pentru a identifica fețele dintr-o imagine, am construit o piramidă Gaussiană de imagini, reducând rezoluția cu un factor de 1.25 la fiecare iterație. Apoi, am aplicat metoda ferestrei glisante cu dimensiunea de 40×40. Fiecare fereastra a fost clasificata de către un *support vector machine* de parametrii C=100, kernel='rbf', gama='scale', aplicând înainte un *Standard Scaler*.

```
Pipeline([('scaler', StandardScaler()), ('svc', SVC(C=100, kernel='rbf', gamma='scale'
    ))])
```

SVM-ul a fost antrenat pe descriptorii HOG parameterii orientations=9, pixels_per_cell = (10, 10), cells_per_block=(4,4), block_norm="L2-hys".

1.2 Generarea exemplelor pozitive

Pentru fiecare imagine din setul de antrenare am decupat fețele , le-am redimensionat la 40×40 și am extras descriptorii HOG cu parametrii de mai sus. Pentru mărirea setului de date am generat descriptorii și pentru fața oglindită vertical.

1.3 Generarea exemplelor negative

În primul rând am salvat o listă cu lungimile și lățimile fiecărui dreptunghi aferent fețelor din setul de antrenare. Pentru exemplele negative, am încercat sa extrag 4 dreptunghiuri random din lista extrasa astfel încât sa nu se intersecteze între ele și nici cu fețele. În final am avut 13954 exemple pozitive și 14964 exemple negative.

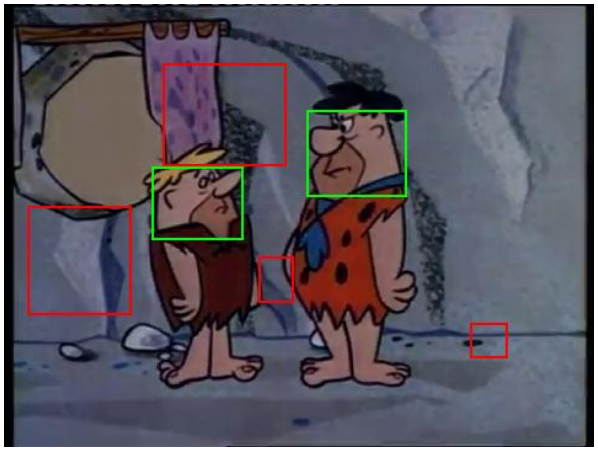


Figura 1: Rosu = patch negativ, Verde = patch pozitiv

1.4 Rezultat

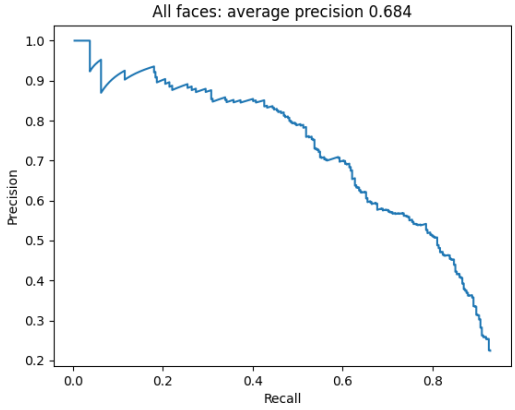


Figura 2: Rezultat înainte de negative mining

1.5 Hard negative mining

Am luat clasificatorul și pentru fiecare imagine din setul antrenare am salvat patch-ul clasificat drept față cu scor maxim care are Intersection over Union maxim 0.3. Am observat că dacă există o detecție mare (D1) care conține o detecție mică (D2) cu

$$score(D1) > score(D2)$$

, astfel încât

$$iou(D1, D2) < 0.3$$

o să fie păstrate amândouă. Pentru a rezolva problema am introdus și intersection over minimum size.

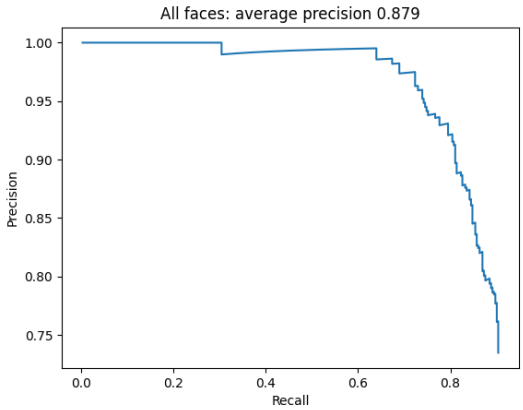


Figura 3: Rezultat după negative mining

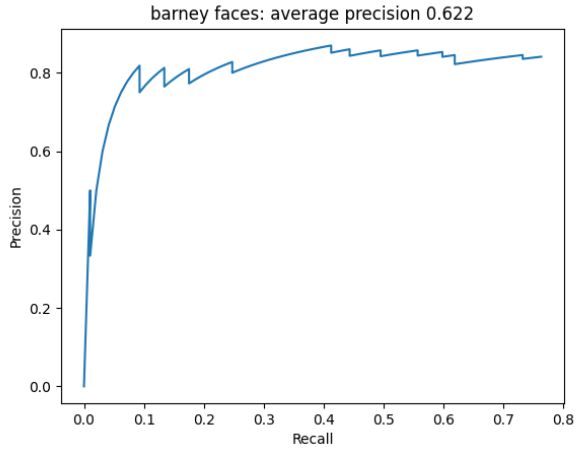
2 Task 2

2.1 Modelul folosit

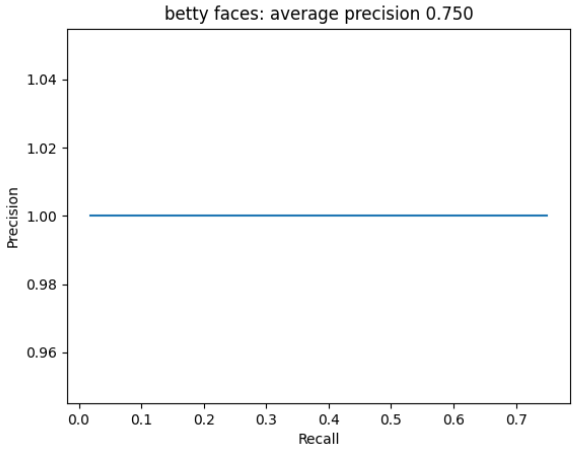
Am antrenat un SVM de parametrii C=1, kernel='rbf', gamma='scale' si decision_function_shape='ovr' pe fețele din setul de antrenare redimensionate la 40×40 și vectorizate, în plus am adăugat si oglindirea verticală a fiecărei decupări. După ce am testat am obținut 0.68 average precision, observând că nu

clasifică bine exemplele de fețe fals pozitive (care de fapt sunt o bucată de background și seamănă cu exemplele de la negative mining) așa că am adăugat și toate patch-urile obținute din negative mining de la task-ul anterior. Astfel am obținut un set de date cu 5 etichete : 'barney', 'betty', 'fred', 'unknown', 'background', iar după antrenare am obținut 0.725 average precision.

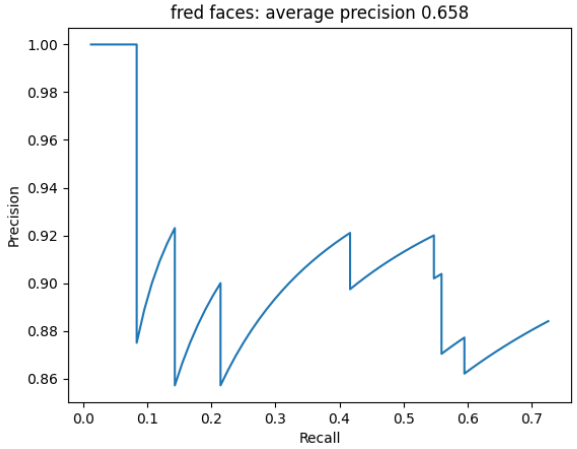
2.2 Rezultat



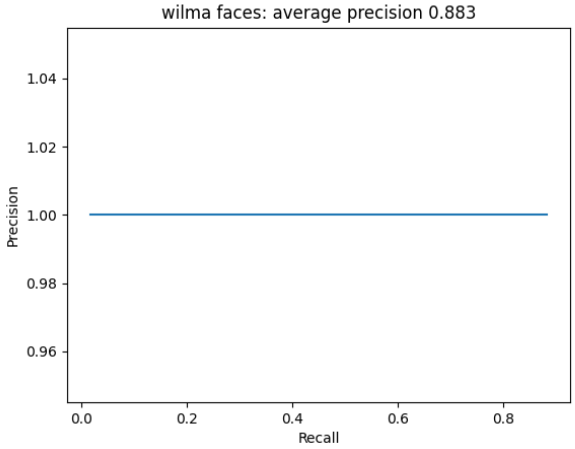
(a) Barney



(b) Betty



(c) Fred



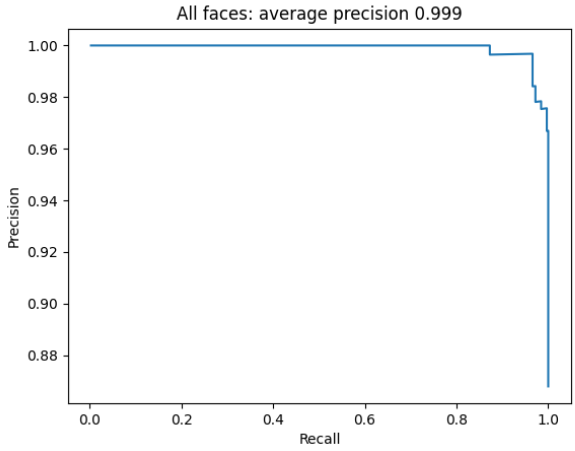
(d) Wilma

3 Bonus

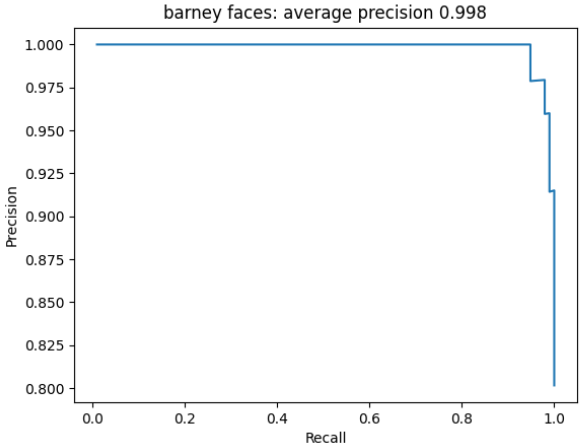
3.1 Modelul folosit

Pentru bonus am folosit arhitectura Faster R-CNN care folosește ResNet50 pentru a extrage caracteristicile și care adaugă Feature Pyramid Network pentru a detecta obiecte la scăli diferite. L-am antrenat de la zero pe setul de antrenare, redimensionând imaginile la 240×240.

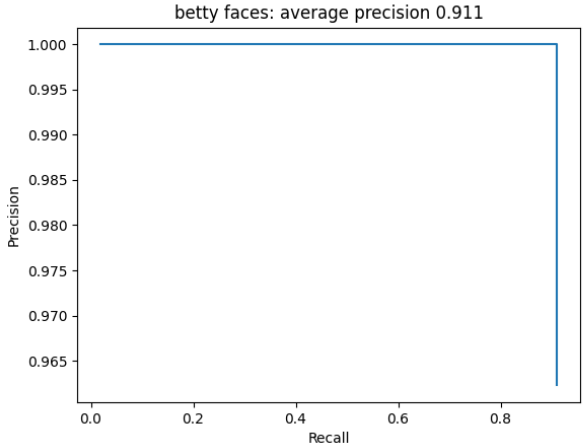
3.2 Rezultat



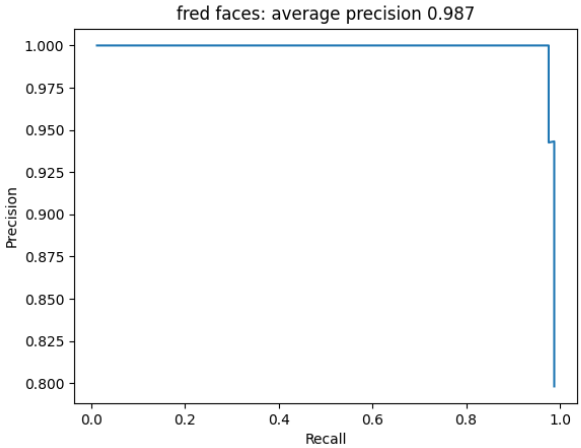
(a) Faster R-CNN all faces



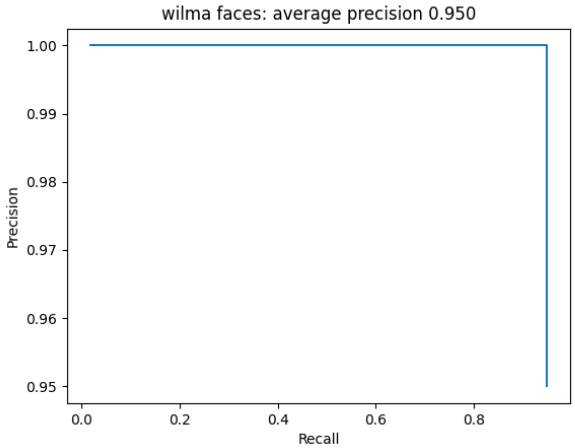
(b) Barney



(c) Betty



(d) Fred



(e) Wilma