Documentatie - Detectare si recunoastere faciala

Ciuperceanu Vlad-Mihai
Grupa 351

1 Introducere

Acest proiect are ca scop detectarea si recunoasterea faciala a personajelor din Laboratorul lui Dexter in imaginile date. Acestea dau directia celor 2 task-uri principale, in timp ce task-ul bonus permite folosirea unui detector state-of-the-art pentru rezolvarea acestor probleme. Pentru primele 2 task-uri vom folosi clasificatori antrenati pe baza descriptorilor HOG, urmand paradigma ferestrei glisante, in timp ce pentru task-ul bonus vom folosi modelul YOLO pentru a obtine rezultate aproape perfecte.

2 Precalculari si observatii

Inainte de a decide parametrii pentru solutiile noastre, putem analiza, mai intai, imaginile din setul de antrenare, pentru a obtine mai multe informatii despre acestea.

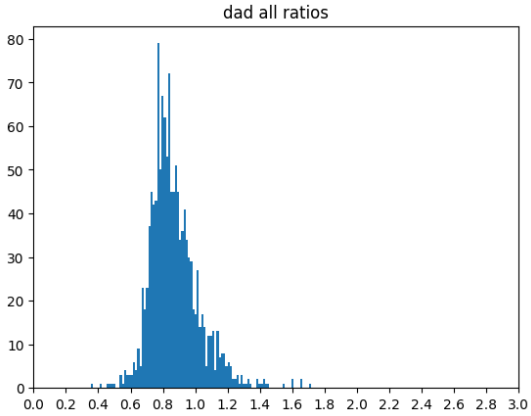
Evident, observam ca fetele variaza destul de mult, atat in dimensiune, cat si in forma (unele personaje avand fetele care pot fi incadrate usor cu un patrat, altele cu un dreptunghi mai lung, altele cu un dreptunghi mai inalt).

Extragand adnotarile pentru fiecare fata, observam ca dimensiunile lor sunt destul de mari, ambele dimensiuni fiind in jur de 150 de pixeli. Acestea merg, bineinteles, si spre valori mai mici, cum ar fi 50-60 pixeli, respectiv mai mari, cum ar fi 200 pixeli. Astfel, obtinem o limita superioara pentru dimensiunea unei ferestre, putand, desigur, sa marim imaginile in anumite situatii.

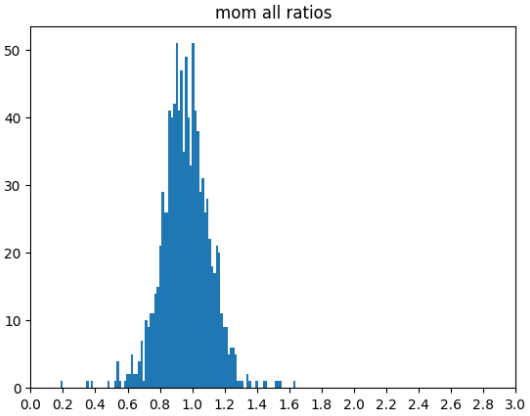
Din moment ce este nevoie de o abordare multi-scale, in care redimensionam imaginea, ne dam seama ca este mai usor de lucrat cu acest aspect si ca alegerea dimensiunii nu este una atat de stricta, putand functiona cu mai multe valori cu diferente destul de mari, calibrand parametrii din abordarea multi-scale.

Totusi, in cadrul redimensionarii, aspect ratio-ul se pastreaza, astfel ca trebuie sa fim atenti la raportul dimensiunilor fetelor. Aceste rapoarte se vor translata in cadrul ferestrelor (asa cum mentionam mai sus, fetele au diferite forme, astfel ca ar fi de ajutor sa folosim mai multe ferestre).

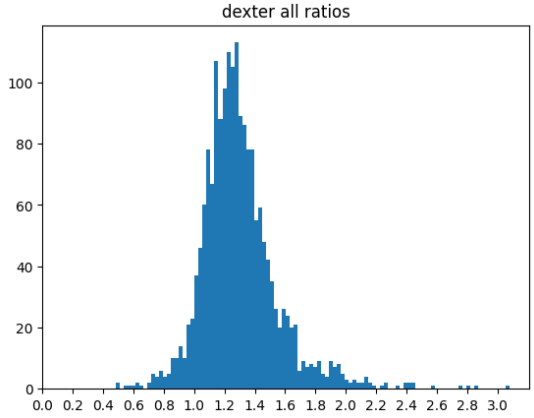
Analizand datele obtinute din adnotari si extragand coordonatele fetelor, putem calcula aspect ratio-ul pentru fata respectiva. Pentru o vizualizare mai buna, am facut cate o histograma pentru cele 4 personaje principale si categoria personajului necunoscut, pentru a vedea mai clar distributia rapoartelor in cadrul fetelor identificate. Am obtinut:



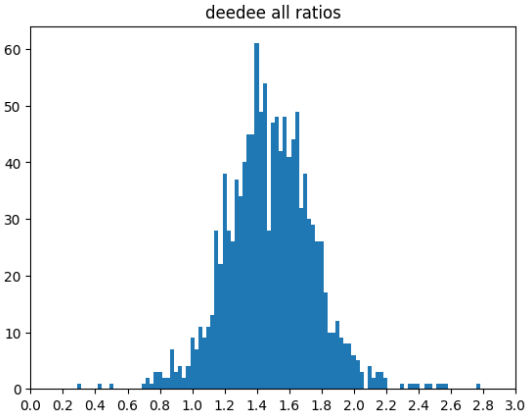
(a) Dad



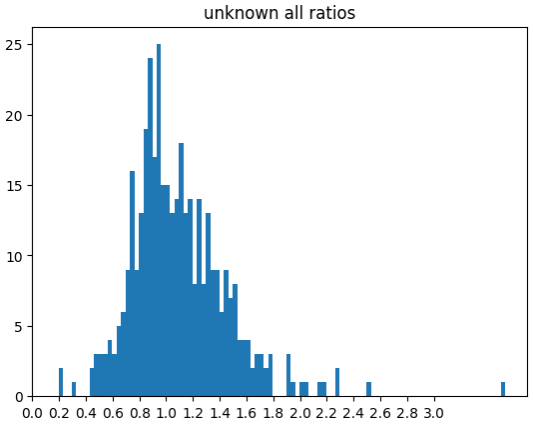
(b) Mom



(c) Dexter



(d) DeeDee



(e) Unknown

Figura 1: Histogramele aspect ratio-urilor personajelor

Analizand histogramele obtinute, observam ca zona cea mai condensata pentru fiecare personaj se afla in intervalele:

- $[0.75 - 0.85]$ pentru Dad
- $[0.9 - 1.1]$ pentru Mom
- $[1.1 - 1.3]$ pentru Dexter
- $[1.4 - 1.6]$ pentru DeeDee
- $[0.9 - 1.1]$ pentru Unknown

Astfel, observand varietatea datelor, am decis sa folosim 4 ferestre pentru 4 aspect ratio-uri diferite: 0.75, 1.0, 1.25, 1.5. Numerele au fost alese pentru a reprezenta zonele condensate din fiecare histograma, punand accent pe cele 4 personaje de interes, din moment ce acestea apar cel mai des. Totodata, am urmarit o distanta echilibrata intre acestea, incercand sa prindem si din fetele incadrate cu patrate, si din cele cu fete dreptunghiulare (atat pe inaltime, cat si pe lungime).

3 Abordare generala

Avand in vedere variatia fetelor personajelor, asa cum am mentionat mai sus, vom folosi 4 tipuri de ferestre pentru a acoperi fetele celor 4 personaje principale, dar si ale celor necunoscute, care pot fi detectate de cel putin una dintre ferestre (pentru task-ul 1).

Ca abordare generala asupra celor 2 task-uri, etapele prin care trecem sunt urmatoarele:

- obtinerea descriptorilor HOG pozitivi si negativi
- antrenarea modelelor pe baza descriptorilor obtinuti
- prezicerea detectiilor, respectiv a clasificarilor pentru fetele gasite

3.1 Numar exemple pozitive si negative

Numarul exemplelor pozitive depinde de numarul de fete existente in folderul de antrenare. Am ales sa dam flip stanga-dreapta, pentru a ne dubla numarul de exemple pozitive, ajungand la peste 11 000 de astfel de exemple. Pentru numarul de exemple negative, am ales valoarea de 48 000, din motivul ca antrenarea modelelor cu kernel RBF este mai costisitoare ca timp de executare, iar un numar mare de exemple negative (si a setului de date de antrenare in general) crestea in mod clar timpul de antrenare si de predictie.

Totodata, nu este de neglijat nici discrepanta dintre numarul de exemple pozitive si cel de exemple negative, care ar putea duce la o invatare slaba si la o lipsa de capacitate de generalizare. Tocmai de aceea, am incercat sa gasim valori care sa ajute modelele in acest proces.

3.2 Dimensiuni ferestre

Vom pastra rapoartele mentionate mai sus, astfel ca este nevoie sa alegem o singura dimensiune, cea de-a doua fiind calculata in functie de aceasta.

Am ales sa pornim de la dimensiunea de 90 pixeli (parametrul dim_window), cu $dim\_hog\_cell = 18$, urmand ca pentru ferestrele dreptunghiulare sa avem latura mai mica de aceasta dimensiune, pastrand rapoartele dorite (e.g. fereastra de aspect ratio 1.5 va avea dimensiunile 90 si 135). Numarul pentru dimensiunea de pornire a ferestrei a fost ales prin multiple incercari, cu numere precum 36, 48, 50, 56, 60, 75, 100, incercand sa gasim ferestre care sa fie potrivite pentru media fetelor din imagini. Desigur, pentru fetele mai mici era nevoie de o micsorare mai mare a imaginii, pornind de la imaginea initiala, pe cand pentru ferestrele mai mari era nevoie de o marire initiala, urmata de micsorari treptate, cu pasi mai mici. Toate aceste reglari de parametri aveau ca scop detectia atat a imaginilor mai mici, cat si a celor mai mari.

Pentru dim_hog_cell am ales sa pastram un raport mai intre dimensiunea ferestrei si cea a celulei, intrucat aceasta avea 2 implicatii: pe de-o parte descriptorii deveneau mai complecsi si cresteau in dimensiune, ceea ce ar fi oferit teoretic o precizie mai buna per

model, dar, pe de alta parte, crestea semnificativ timpul de invatare necesar, precum si cel de predictie, intrucat modelul avea nevoie sa fie mult mai complex pentru a gasi separarea corecta. Pentru modelele cu kernel liniar, ne permiteam sa incercam, de exemplu, valori de tipul $dim\_window = 5$ si $dim\_hog\_cell = 5$, insa, pentru kernel RBF, un astfel de raport, de 15, ingreuna lucrul cu modelele.

3.3 Modele

Modelele folosite corespund fiecarui tip de fereastra, astfel ca avem 4 modele.

Tipul modelului folosit este SVC cu kernel RBF, setand parametrul C la 1:

```
model = SVC(C=1, kernel='rbf', verbose=True)
```

Verbose a fost setat cu $True$ pentru a urmari progresul antrenarii modelului. Desigur, acesta avea nevoie de un timp mai mare de antrenare decat cel cu kernel liniar, durand intre 10 si 20 de minute. Parametrul C a fost setat la 1 pentru a ne apropia mai mult de o convergenta a modelului, aceasta valoare fiind mai convenabila. Thresholdul pastrat a fost unul de 0, observand ca precizia medie este destul de sensibila la modificarea thresholdului, chiar si a unuia mic, ducand la o scadere de la 1-2% pana la 5-6%.

Anterior, am incercat si modele SVC cu kernel liniar, insa am observat un boost de pana la 20% in schimbarea kernelului.

Aceeasi paradigma de modele a fost folosita atat pentru task-ul 1, cat si pentru task-ul 2, diferenta fiind facuta de felul in care acestea au fost antrenate, adica de alegerea exemplelor pozitive si negative.

3.4 Detalii despre prezicere

Pentru partea de prezicere, avand in vedere ca fetele pot aparea la diferite dimensiuni (mai mici sau mai mari, in functie de pozitia personajelor in cadru), folosim o abordare multi-scale, redimensionand imaginea pentru a detecta mai bine fetele, astfel creand o piramida de imagini. Initial marim putin fata, apoi incepem sa o micsoram pana cand dimensiunea ei devine mai mica decat cea a ferestrei cu care lucram.

Pentru crearea piramidei de imagini, am folosit functia *pyramid_gaussian* din scikit-image, care genera imagini pornind de la o imagine data, facand downscale cu un factor dat si aplicand filtre gaussiene pentru netezire:

```
for resized_image in pyramid_gaussian(img, downscale=1.05)
```

Factorul ales este unul de 1.05 pentru a nu scadea foarte repede dimensiunile imaginii, avand in vedere ca folosim ferestre destul de mari. Atunci cand obtinem o detectie avem grija sa o rescalam la coordonatele corespunzatoare in imaginea initiala, tinand cont de gradul de downscaling de pana acum, cat si de posibilele redimensionari (mai exact, mariri) initiale.

Desigur, urmam acesti pasi cu fiecare fereastra folosita, urmand sa aplicam non-maximum suppression intre acestea pentru a filtra detectiile obtinute pe imagine.

Acum ne vom axa pe detaliile ce tin de fiecare task in parte.

4 Task 1 - Detectie

Pentru primul task, modelele noastre aveau nevoie sa faca detectia intre fata si non-fata. Folosirea mai multor modele este folosita pentru a detecta mai multe tipuri de fete, combinand rezultatele (fiecare model fiind antrenat cu descriptori pozitivi si negativi pentru tipul de fereastra pe care il reprezinta). Am ales sa unim rezultatele aplicand non-maximal suppression asupra detectiilor obtinute pe o imagine pentru a le pastra doar pe cele mai bune.

4.1 Descriptori pozitivi

Descriptorii pozitivi au fost alesi din fetele adnotate.

Se poate observa ca fetele sunt, uneori, adnotate mai strans, astfel ca am considerat patch-ul fetei putin extins, adaugand un extra 10% din dimensiunea fetei pe ambele dimensiuni. Apoi, din patch-ul obtinut a fost calculat descriptorul HOG care a fost adaugat la lista de descriptori pozitivi.

Totusi, imbalanta dintre numarul de exemple pozitive si cel negative ramane, fiind important, desigur, ca modelele sa aiba un numar mare de exemple de antrenare pentru a capata o capacitate de generalizare mai buna.

Astfel, am incercat sa crestem numarul cu exemple pozitive folosind o maniera folosita in cadrul prezicerii unei fete: construirea unei piramide gaussiene. Preluand patch-ul initial, dar folosind un downscale mai agresiv, de 1.2, am redimensionat patch-ul respectiv, urmand sa calculam descriptorul HOG din redimensionarea acestuia. In acest mod am putut creste numarul de exemple pozitive, adaugand si unele exemple mai robuste la variatiile din imagini datorita filtrului aplicat.

Aceasta adaugare de exemple pozitive a adus o crestere de 5-10% in precizia medie obtinuta.

4.2 Descriptori negativi

Pentru exemplele negative a fost nevoie sa alegem patch-uri aleatoare din imagini, incercand sa evitam fetele, lasand o margine mica de 0.15 pentru calculul functiei intersection over union intre acestea.

Totodata, a fost de ajutor sa alegem patch-uri cu dimensiuni variabile, in care pastram ratio-ul pentru fereastra care ne interesa si foloseam *cv.resize* dupa, in loc sa selectam mereu aceleasi dimensiuni pentru patch-uri. In acest fel, introduceam mai multa varietate in exemplele negative alese, reducand din posibilitatea overfitting-ului si aducand o crestere de 3-5% in precizia medie obtinuta.

4.3 Rezultate

Cu abordarea descrisa mai sus, am obtinut urmatoarea precizie medie:

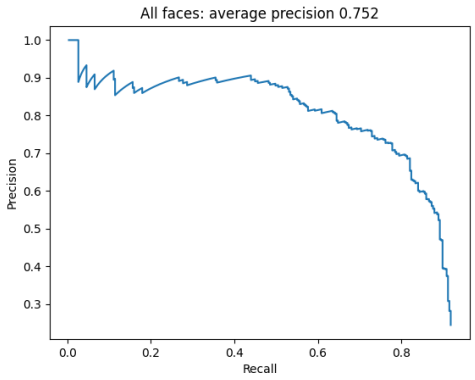


Figura 2: Precizie medie pentru detectii

5 Task 2 - Clasificare

Pentru al doilea task, modelele nu mai reprezentau doar dimensiunea ferestrei, ci si personajul detectat. Astfel, fiecare din cele 4 modele erau cate un detector de personaj, avand atribuit personajul din a carui histograma a fost ales aspect ratio-ul ferestrei.

Modalitatea de prezicere ramane aceeasi ca in cazul primului task, folosind multi-scale. O mica ajustare este aceea ca, prin incercari repetate, am observat ca, pentru DeeDee, nu ajuta marirea imaginii inainte de crearea piramidei gaussiene, spre deosebire de celelalte personaje (pierzand 2% din precizia medie). Acest lucru poate fi datorat dimensiunii ferestrei, care este cea mai ridicata (aspect ratio de 1.5).

Singura diferenta majora este ca, in acest caz, fiecare aducea rezultate in mod independent, nemaifiind nevoie sa combinam rezultatele, din moment ce eram interesati de detectiile fiecarui personaj in parte.

5.1 Descriptori pozitivi

Exemplele pozitive au fost alese in mod asemanator cu cele de la primul task, facand o mica marire cu factor de 1.2 asupra patch-ului personajului, diferenta fiind ca fiecare clasificator folosea ca exemple pozitive doar fetele personajului sau, nu pe toate.

5.2 Descriptori negativi

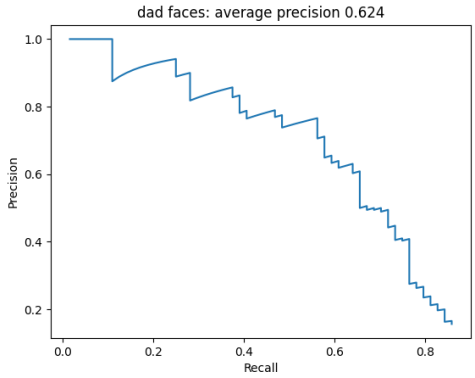
Avand nevoie de o clasificare de tipul Dexter vs non-Dexter, era nevoie ca modelul folosit sa poata face diferenta intre aceste clase.

Am impartit aceasta capacitate in 2: de a face diferenta intre un personaj si celelalte personaje si de a face diferenta intre o fata si o non-fata.

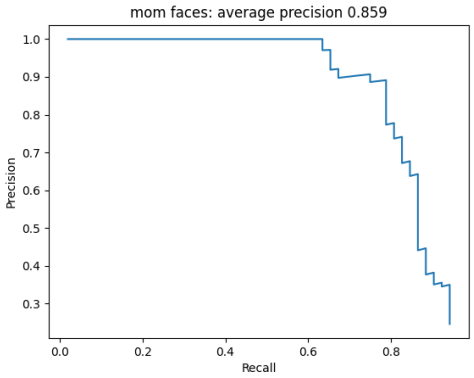
Astfel, numarul de exemple negative a fost impartit in 2: jumatate dintre ele au fost data de descriptorii HOG ale patch-urilor celorlalte personaje, iar cealalta jumatate fiind data de patch-uri random care sa nu aiba intersection over union prea mare cu fetele.

5.3 Rezultate

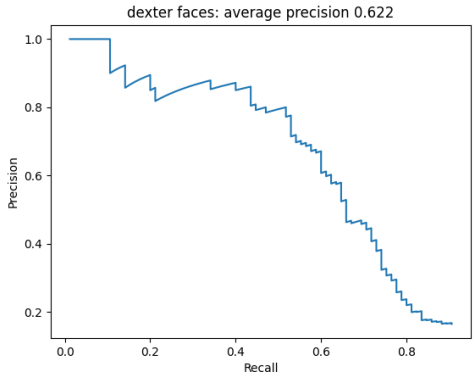
Folosind abordarea descrisa mai sus, am obtinut urmatoarele precizii medii:



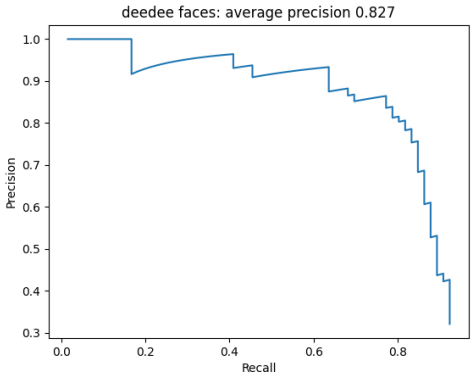
(a) Dad



(b) Mom



(c) Dexter



(d) DeeDee

Figura 3: Precizia medie pentru fiecare personaj

6 Task Bonus - YOLO

Pentru task-ul bonus, am ales folosirea modelului YOLOv8.

Modul in care acesta a fost folosit este urmatorul:

- procesarea datelor de antrenare, pentru a corespunde formatului necesar al modelului
- antrenarea modelului cu datele noastre de antrenare
- preluarea modelului obtinut si testarea pe imagini
- procesarea output-ului si convertirea inapoi la formatul nostru

6.1 Procesarea input-ului

In partea de antrenare a modelului, este nevoie de:

- fisierele organizate in folderul */datasets/train*, care contine folderele */images* si */labels*

- un fisier .yaml in care sa precizam caile catre folderele de antrenare, validare, testare, precum si numarul de clase si label-urile lor

Adnotarile trebuie, de asemenea, sa respecte formatul impus de model, sub forma: *class_id x_center y_center width height*, date normalizate, fata formatul anterior de

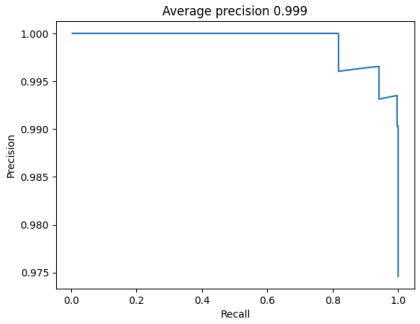
adnotari, in care foloseam denumirea personajului si coordonatele x_min y_min x_max y_max.

6.2 Procesarea output-ului

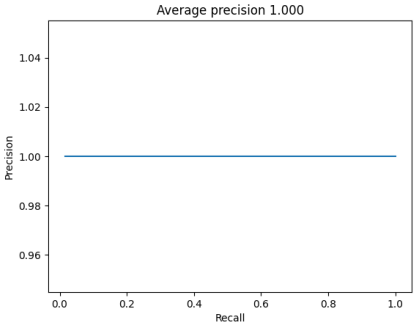
Dupa obtinerea predictiilor, acestea trebuie procesate din nou, in mod invers fata de input, pentru a ne intoarce la formatul initial. Adnotarile obtinute se afla in folderul */labels* din folderul precizat. In final, putem salva predictiile obtinute ca la celelalte task-uri.

6.3 Rezultate

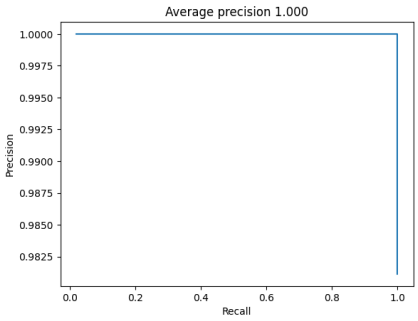
Rezultatele obtinute de acest model sunt urmatoarele:



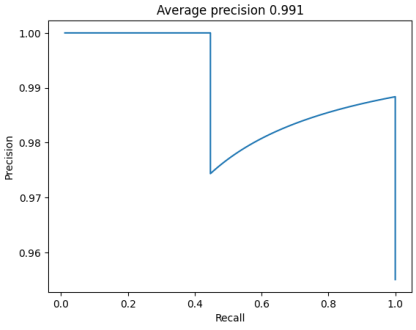
(a) Detectare de fete



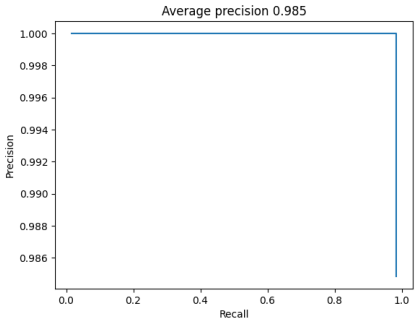
(b) Dad



(c) Mom



(d) Dexter



(e) DeeDee

Figura 4: Precizii medii YOLO