Documentație Pentru Tema 2 Computer Vision - Scooby-Doo

Rizea Mihai-Marius

18 ianuarie 2026

Cuprins

1 Introducere

În realizarea acestei teme am pornit de la codul pus la dispoziție în cadrul Laboratorului 9+10. În această documentație sunt prezentate modificările și contribuțiile personale aduse, precum și rezultatele obținute.

2 Secțiunea 1: Crearea setului de date

În etapa inițială, a fost construit un set de date format din 6547 exemple pozitive. Pornind de la adnotările ground-truth ale celor 4000 de imagini, a fost implementat un script care extrage fiecare bounding box corespunzător și îl redimensionează la dimensiunea standard 36×36 pixeli.

Pentru exemplele negative, a fost creat manual un set de date cât mai divers și reprezentativ. Acesta include atât obiecte cu forme rotunde, care pot fi ușor confundate cu o față, cât și imagini cu personajul Scooby, pentru care au fost adăugate intenționat mai multe exemple negative, cu scopul de a reduce confuziile și de a împiedica modelul să îl detecteze eronat.

Pentru a avea un număr mai mare de exemple pozitive, mi-am creat în funcția de exemple pozitive noi metode de augmentare, precum rotirea patch-ului de la -5 la 5 grade, adaptare de luminozitate sau de contrast (modificând canalele alfa și beta).

Pentru funcția de extragere de exemple negative, care în laborator funcționa pentru exemple strict mai mari de 36px, atât pe lungime cât și lățime, am adăugat 2 noi cazuri: cel în care exemplul este de fix 36×36px (în care extrage automat descriptorul Hough aferent), dar și pentru cele mai mici, în care face resize la 36×36 și apoi extrage descriptorul.

2.1 Modificări aduse SVM-ului

În funcția de `train_classifier`, doar la Task 2 am modificat modelul, punând parametrul `class_weight='balanced'`, deoarece am mult mai multe exemple negative decât pozitive.

2.2 Modificări aduse NMS

În funcția de `non_maximal_suppression`, pragul de suprapunere a fost redus de la 0.3 la 0.15, pentru a elimina cat mai multe posibile fals pozitive, care s-ar putea suprapune peste fata corecta, care are scorul local maxim.

2.3 Paradigma de Fereastra Glisanta

Paradigma de fereastră glisantă am folosit-o în funcția de `run_task`, în care, din imaginea mare, fac un resize la fiecare iterație de 0.9, până când imaginea ajunge cât este fereastra de 36×36. Astfel, nu am nevoie de mai multe ferestre pentru a căuta personajele.

3 Task 2: Procesarea Exemplelor Negative

Pentru Task-ul 2, în cadrul creării fiecărui model pentru fiecare personaj, am considerat ca exemplu pozitiv fețele personajului curent, iar ca exemple negative: celelalte trei personaje, toate personajele din clasa `unknown`, precum și exemplele negative din Task 1, cele cu background-ul.

4 Bonus YOLO

În realizarea bonusului, am luat toate cele 4000 de poze într-un singur folder (pe care le-am redenumit de la 1 la 4000), precum și adnotările acestora într-un singur fișier `.txt`, pe care le-am formatat la standardul YOLO. Antrenarea a fost realizată pe parcursul a 100 de epoci și a durat aproximativ două ore.

5 Rezultatele obținute

În această secțiune sunt prezentate rezultatele experimentelor pentru Task 1 și Task 2, sub formă de grafice de tip Precision–Recall.

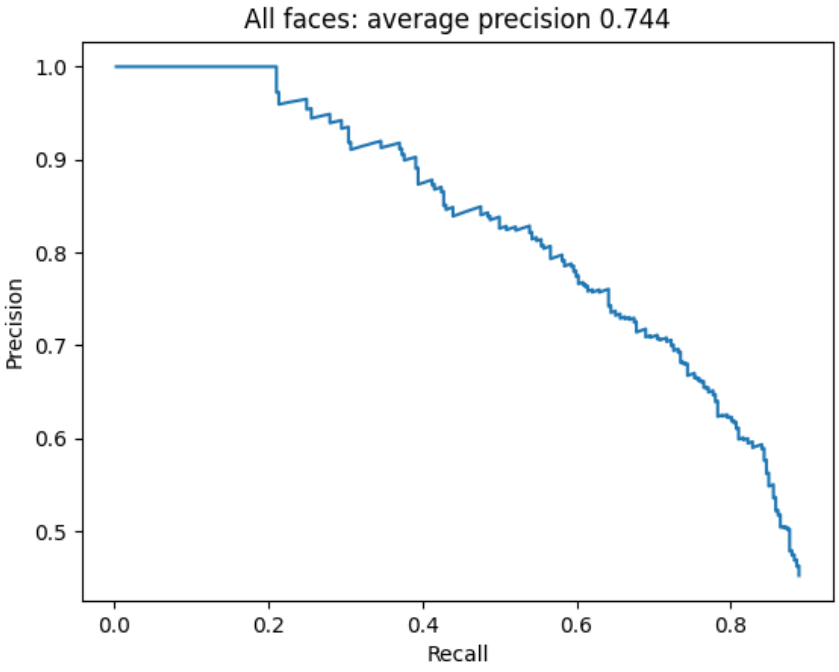
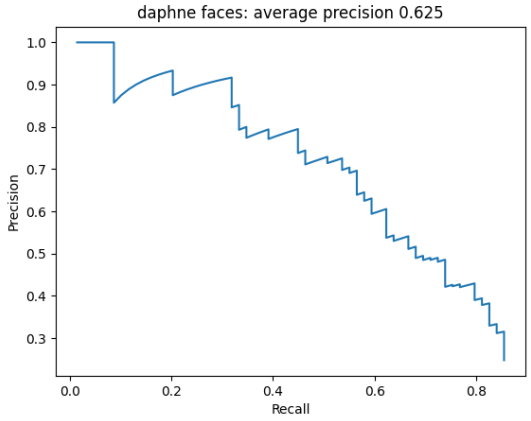
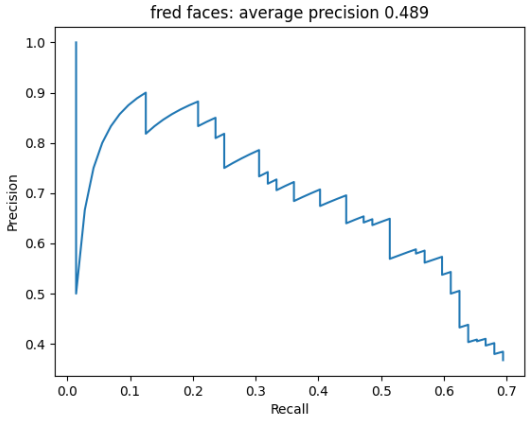


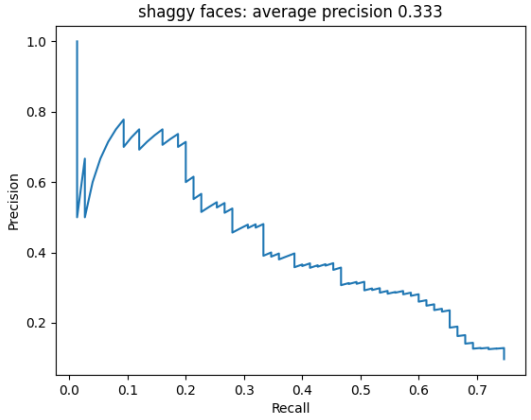
Figura 1: Task 1: Grafic Precision–Recall pentru toate fețele.



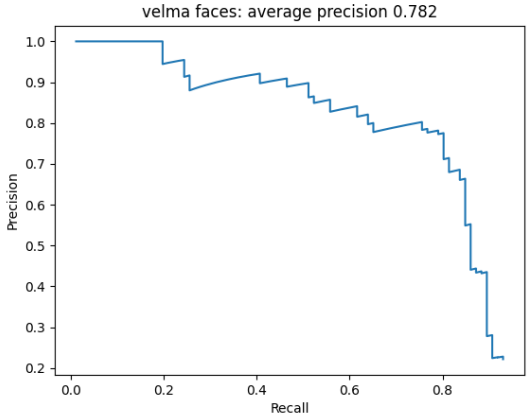
(a) Daphne



(b) Fred



(c) Shaggy



(d) Velma

Figura 2: Task 2: Graficele Precision–Recall pentru fiecare personaj.

5.1 Rezultatele obtinute prin YOLO

În această secțiune sunt prezentate rezultatele pentru Task 1 și Task 2 BONUS, sub formă de grafice de tip Precision–Recall.

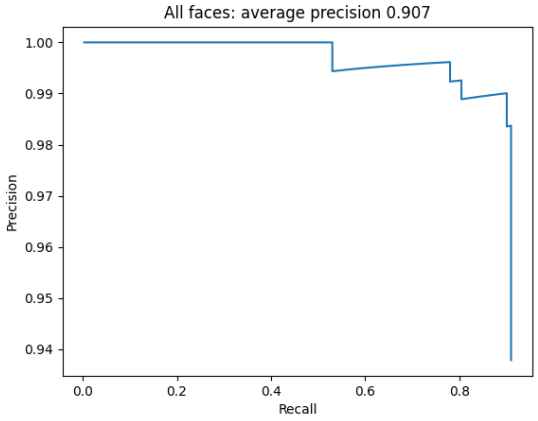
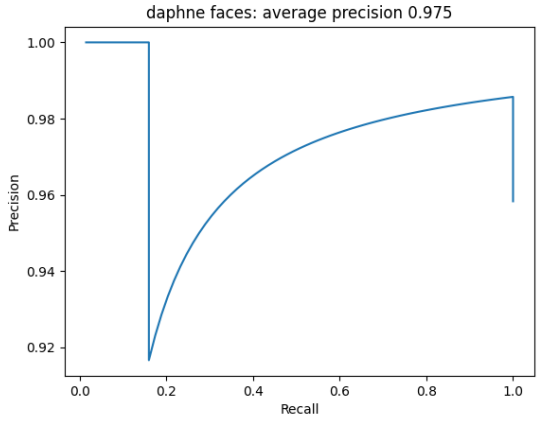
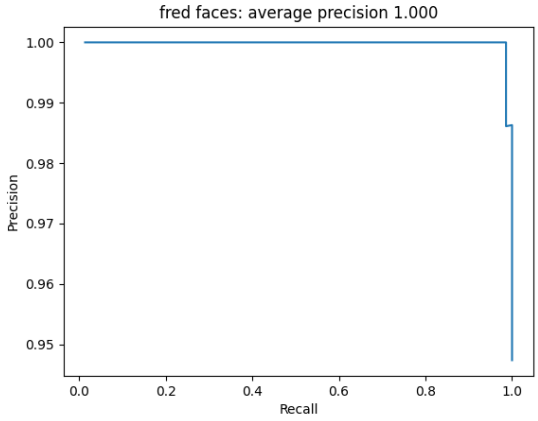


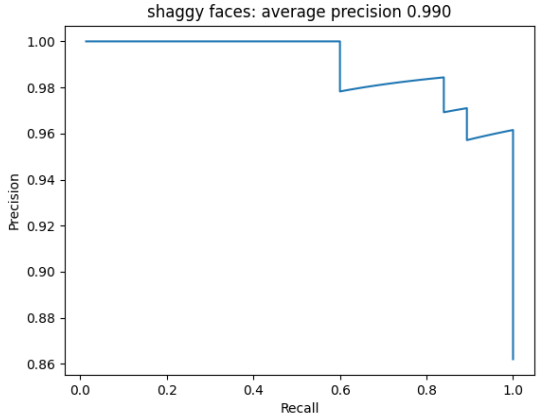
Figura 3: Task 1: Grafic Precision–Recall pentru toate fețele.



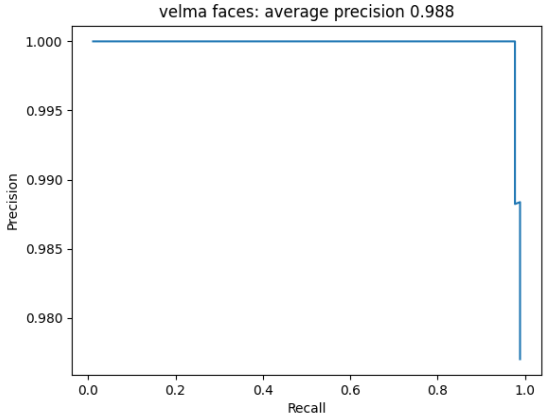
(a) Daphne



(b) Fred



(c) Shaggy



(d) Velma

Figura 4: Task 2: Graficele Precision–Recall pentru fiecare personaj.

6 Concluzii

6.1 Rularea Scriptului

Pentru a rula cu folderul de testare:

- trebuie modificată linia 8 din folderul `Bonus` din `generare_rezultate_finale.py`
- trebuie modificate liniile 9 și 10 din `Parameters.py` din Task 1
- trebuie modificate liniile 11–16 din `Parameters.py` din Task 2

Pentru a rula scriptul, trebuie sa fiti in directorul taskului pe care vreti sa il rulati, si sa scrieti:

```
python RunProject.py
```

După ce ați rulat în ambele foldere scriptul, veți avea în folderul principal un folder numit `fisiere_solutie`, acesta trebuie pus în folderul de evaluare.

Iar pentru a rula bonusul, trebuie sa fiti in directorul Bonus, si sa rulati:

```
python generare_rezultate_finale.py
```

Atat array urile npy pentru taskurile normale cat si bonus se vor afla in folderul principal in `fisiere_solutie`.

Țin să precizez că descriptorii și toate modelele create de mine se află în fișierul trimis.

Dacă doriți să rulați de la 0, trebuie să intrați în `data`, în folderul `salveazaFisiere`și să le ștergeți.

Timpul de rulare este de aproximativ 5 minute pe un laptop cu Intel i5 12500H, placă video RTX 3050 4 GB și 16 GB RAM.

Biblioteci Utilizate

Pentru implementarea și rularea scriptului au fost utilizate mai multe biblioteci Python. Majoritatea fac parte din biblioteca standard Python și nu necesită instalare separată:

- `argparse` – pentru parsarea argumentelor din linia de comandă;
- `sys` – pentru interacțiunea cu interpretorul Python;
- `os` – pentru manipularea căilor și directoarelor;
- `glob` – pentru obținerea listelor de fișiere.

În plus, scriptul utilizează două biblioteci externe. Versiunile folosite în proiectul de față sunt:

- **OpenCV (cv2):** 4.11.0
- **NumPy:** 2.2.3

- **Matplotlib:** 3.10.8
- **Scikit-learn:** 1.8.0
- **Torch:** 2.7.1+cu118
- **Ultralytics (YOLO):** 8.3.248

Scriptul a fost testat și rulat folosind **Python 3.10.9**.