Documentatia pentru tema 2 - Detectarea si recunoasterea faciala a personajelor din serialul de desene animate Scooby-Doo

1. Introducere

Acest proiect documenteaza dezvoltarea unui sistem de viziune artificiala, capabil sa localizeze si sa identifice automat fetele personajelor principale din serialul de desene animate **"Scooby-Doo"**. Proiectul abordeaza doua probleme fundamentale in computer vision: **detectia de obiecte** (Task 1) si **recunoasterea/clasificarea obiectelor** (Task 2). Pentru task 1, personajele vizate sunt toate cele umane, iar, pentru task 2, doar Fred, Daphne, Shaggy si Velma.

Sistemul a fost proiectat sa indeplineasca urmatoarele cerinte tehnice:

- **Task 1 (Localizare):** Identificarea tuturor fetelor prezente intr-o imagine data, indiferent de personaj, returnand coordonatele ferestrelor de incadrare si un scor de incredere.
- **Task 2 (Recunoastere):** Atribuirea fiecarei fete detectate unuia dintre cele patru personaje principale.
- **Bonus (Deep Learning):** Implementarea unei solutii bazate pe retele neuronale convolutionale moderne (**YOLOv8**) pentru a depasi limitarile metodelor clasice.

Proiectul utilizeaza doua paradigme distincte pentru a demonstra evolutia tehnicilor de viziune artificiala:

- **Abordarea clasica (HOG + SVM):** Se bazeaza pe extragerea de trasaturi geometrice utilizand descriptorul **Histogram of Oriented Gradients (HOG)** si clasificarea acestora printr-un model **Linear Support Vector Machine (SVM)**. Localizarea in imagine se realizeaza printr-o strategie de tip sliding window aplicata pe o piramida de imagini pentru a asigura invarianta la scara.
- **Abordarea moderna (YOLOv8):** Utilizeaza un model de detectie din familia **YOLO**, antrenat prin procesul de fine-tuning pe setul de date furnizat. Aceasta metoda a fost optimizata folosind accelerare hardware (GPU) pentru a obtine o precizie superioara si un timp de executie redus.

2. Task 1 – Detectarea faciala

In cadrul acestui task, am implementat un pipeline clasic de detectie, bazat pe extragerea trasaturilor robuste si clasificarea lor supervizata.

2.1 Reprezentarea trasaturilor: Descriptorul HOG

Pentru a reprezenta structura geometrica a fetelor, am utilizat descriptorul **Histogram of Oriented Gradients (HOG)**, care este ideal pentru detectia obiectelor cu structura fixa. Am utilizat doua ferestre de dimensiuni 64x64, respectiv 60x80 pixeli.

2.2 Antrenarea clasificatorului SVM

Am utilizat un model **Linear Support Vector Machine (LinearSVC)** pentru a invata hiperplanul care separa fetele de fundal. Patch-urile extrase conform adnotarilor, redimensionate la dimensiunea fiecarei ferestre reprezinta exemplele pozitive. Patch-urile extrase aleatoriu din regiuni de fundal (care nu se suprapun cu fetele) reprezinta exemplele negative. Raportul utilizat a fost de 1:5 (pozitive:negative) pentru a evita supra-specializarea.

2.3 Strategia de detectie: Sliding Window si Image Pyramid

Deoarece fetele pot aparea la dimensiuni diferite in cadrul imaginilor de test, am implementat **Image Pyramid** si **Sliding Window**. Imaginea originala este redimensionata succesiv cu un factor de 0.9 pana la atingerea dimensiunii minime a ferestrei. Pe fiecare nivel al piramidei, o fereastra glisanta parcurge imaginea cu un pas fix. Pentru fiecare pozitie, se extrage descriptorul **HOG** si se interogheaza modelul **SVM**.

2.4 Post-procesare: Non-Maximum Suppresion (NMS)

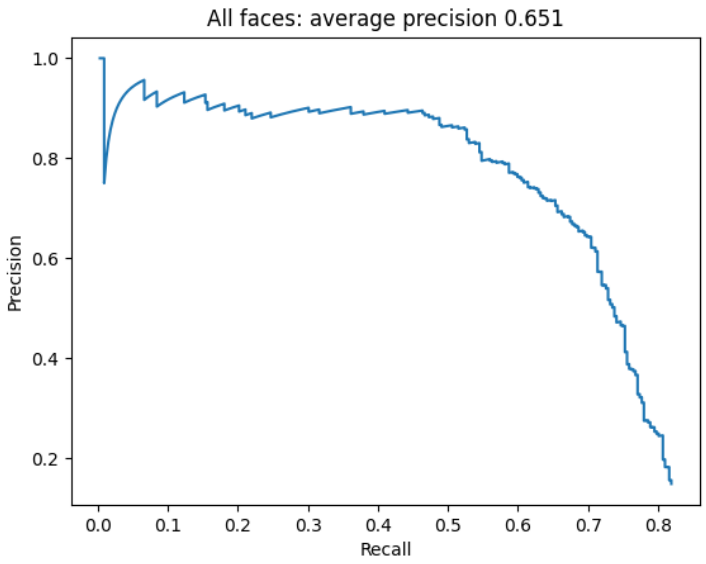
O problema inerenta ferestrei glisante este generarea de detectii multiple (suprapuse) in jurul aceleiasi fete. Pentru a retine o singura fereastra per fata, am implementat algoritmul **Non-Maximum-Suppresion**. Fereastra cu cel mai mare scor este pastrata si o numim „predominanta”, iar celelalte ferestre care au o suprapunere (**Intersection over Union – IoU**) mai mare de pragul 0.3 sau care au centrul in interiorul ferestrei predominante sunt eliminate.

2.5 Salvarea rezultatelor

Informatiile despre detectiile, scorurile si numele fisierelor rezultate in urma pasilor de mai sus au fost salvate in directorul **333_Coman_IrinaElena/task1**, in fisierele **detections_all_faces.npy**, **scores_all_faces.npy**, respectiv **file_names_all_faces.npy**.

2.6 Precizia medie

Cu aceasta solutie, am obtinut un **Average Precision (AP)** de 65.1%.



Rezultatul pentru Task 1

3. Task 2 – Recunoasterea personajelor

Obiectivul acestui task este identificarea identitatii fetelor detectate anterior. Desi localizarea ramane aceeasi, componenta de clasificare devine una multi-clasa, vizand cele patru personaje: **Fred, Daphne, Shaggy** si **Velma**.

3.1 Organizarea setului de date multi-clasa

Spre deosebire de Task 1, unde etichetele erau binare, pentru acest pas am extras patch-uri pozitive separat pentru fiecare personaj, utilizand fisierele de adnotari. Clasele vizate sunt: 0 – Fred, 1 – Daphne, 2 – Shaggy si 3 – Velma. Am extras un numar egal de exemple pentru fiecare personaj (1000 patch-uri per personaj) pentru a preveni bias-ul

clasificatorului catre un anumit erou. Am pastrat o categorie de „fundal” si „personaje secundare” pentru a asigura ca modelul nu forteaza atribuirea unui personaj principal unei fete necunoscute.

3.2 Arhitectura clasificatorului: One-vs-Rest (OvR)

Am ales sa utilizez strategia **One-vs-Rest** cu modele **Linear SVM**. Aceasta abordare presupune antrenarea a patru clasificatori binari independenti, cate unul pentru fiecare personaj. De exemplu, pentru SVM Fred, avem Fred (+1) si {Daphne, Shaggy, Velma, Unknown, Fundal} (-1).

3.3 Procesul de predictie si Regula „Max-Score”

In timpul etapei de testare, pentru fiecare fereastra extrasa si reprezentata prin **HOG**, am interogat toate cele patru modele **SVM.** Fiecare model returneaza un scor, ce reprezinta distanta fata de hiperplanul de separare. Identitatea personajului este data de modelul care returneaza scorul cel mai mare. Pentru a evita clasificarile eronate pe fundal, am impus un prag minim de incredere. Daca niciunul dintre cele patru modele nu depaseste pragul, fereastra este ignorata.

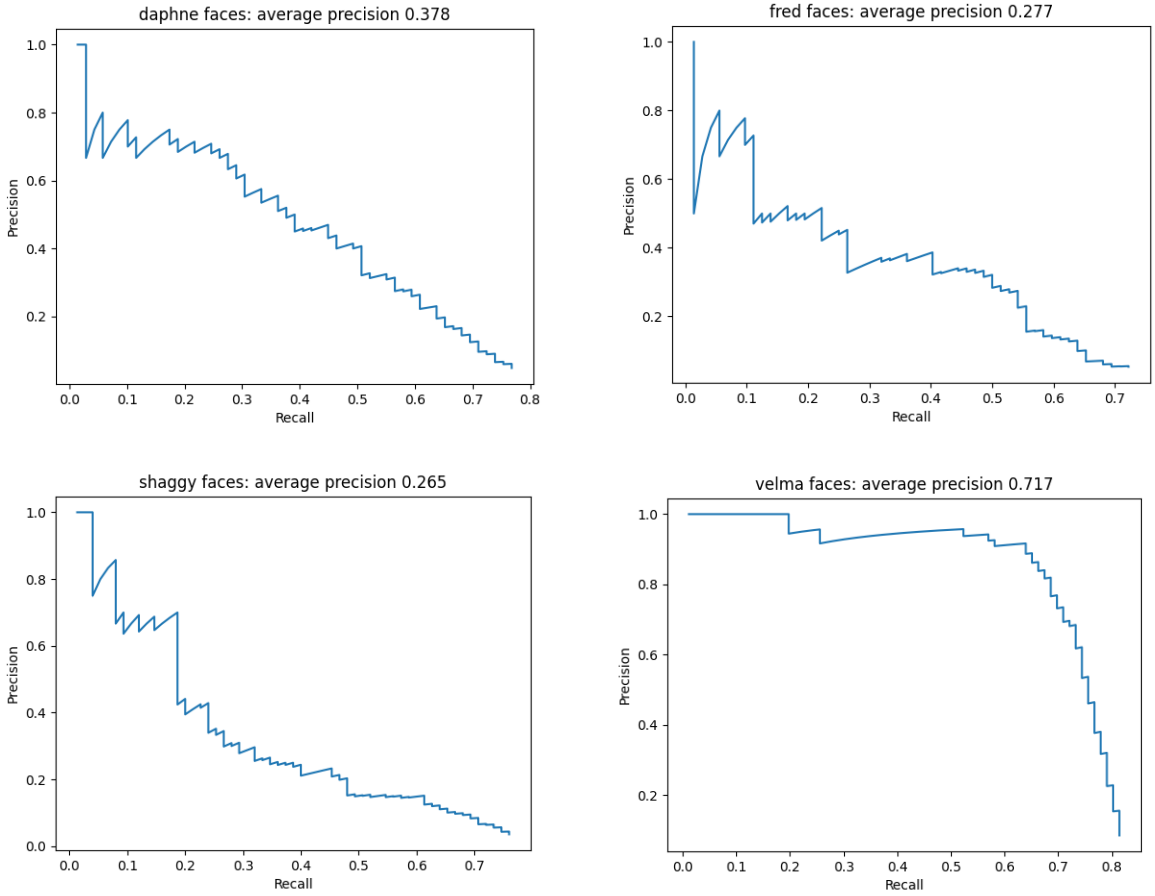
3.4 Salvarea rezultatelor

Informatiile despre detectiile, scorurile si numele fisierelor gasite au fost salvate in acelasi format ca in Task 1, doar ca pentru fiecare personaj. Astfel, fiecarui personaj ii sunt atribuite cate trei fisiere: **detections_personaj.npy**, **scores_personaj.npy** si **file_names_personaj.npy**, ce contin informatii legate doar de acel personaj.

3.5 Precizia medie

Cu aceasta solutie, am obtinut un **Average Precision mediu** de 40.92%, iar pentru fiecare personaj am obtinut:

- **Daphne**: 37.8% AP
- **Fred**: 27.7% AP
- **Shaggy**: 26.5%AP
- **Velma**: 71.7% AP



Rezultatele pentru Task 2

4. Solutia Bonus: Detectie si recunoastere bazata pe Deep Learning (YOLO)

Pentru a depasi limitarile inerente metodelor clasice, am implementat o solutie bazata pe arhitectura **YOLOv8 (You Only Look Once).**

4.1 Pregatirea datelor si augmentarea

Tranzitia catre **YOLO** a necesitat o restructurare completa a setului de date.

- **Conversia adnotarilor**: coordonatele de tip [**xmin, ymin, xmax, ymax**] au fost transformate in formatul **YOLO** [**class, xcenter, ycenter, width, height**], normalizate fata de dimensiunile imaginii.
- **Organizarea fisierelor**: am dezvoltat un script de automatizare pentru redenumirea si centralizarea celor 4000 de imagini de antrenare intr-o structura

compatibila: **images/train** si **labels/train**. De asemenea, acest script muta si imaginile de validare/testare in directorul **images/val**.

4.2 Antrenarea pe Infrastructura Cloud (Google Colab)

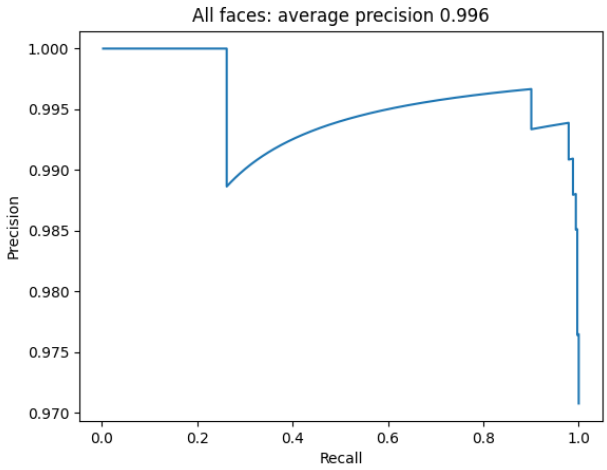
Datorita complexitatii matematice a retelelor neuronale, am utilizat mediul **Google Colab**, beneficiind de accelerare hardware prin **GPU Tesla T4**. Modelul a fost antrenat pe o durata de 50 de epoci, cu o dimensiune a imaginii de intrare de 640x640 pixeli.

4.3 Salvarea rezultatelor

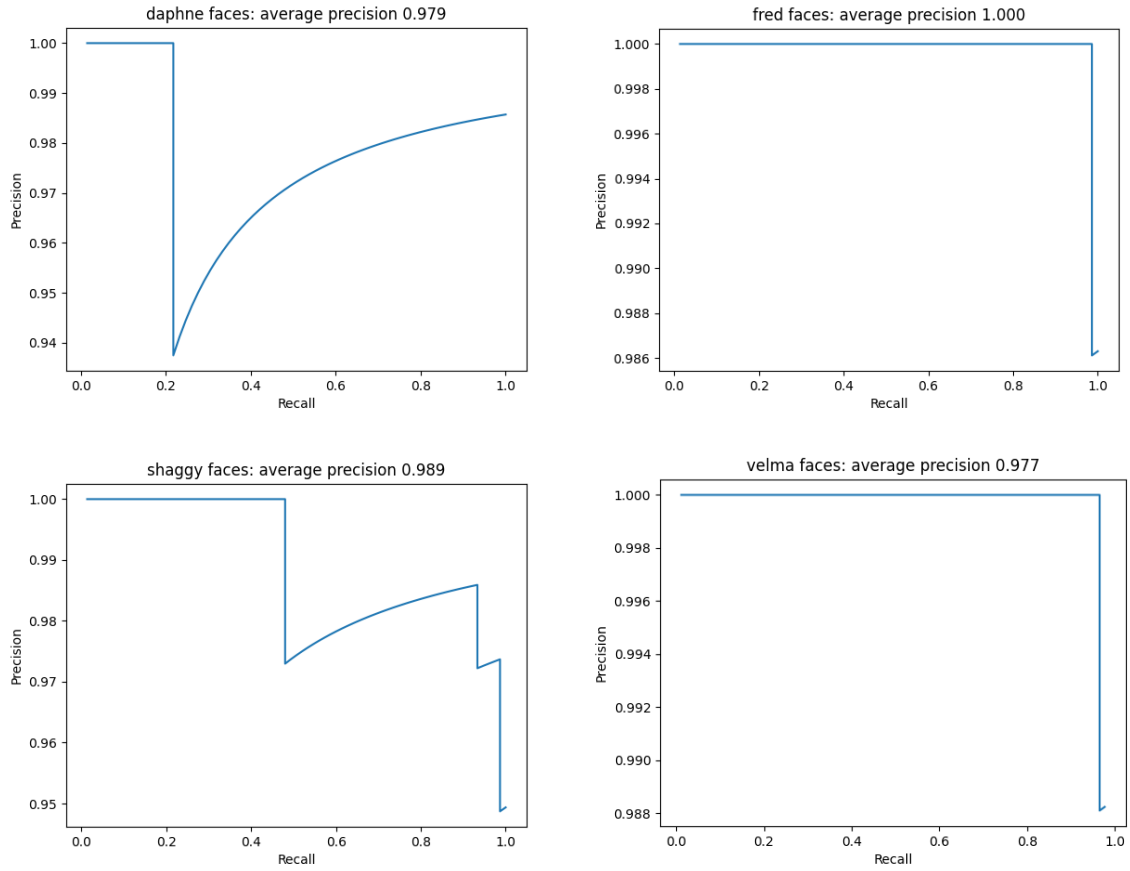
In urma antrenarii modelului pe **Google Colab**, s-a generat un fisier **best.pt**, care a fost incarcat ulterior si folosit pentru executarea detectiei in cadrul proiectului local. Rezultatele obtinute au fost salvate sub acelasi format ca in Task 1 si Task 2, in fisiere **.npy** corespunzatoare.

4.4 Precizia medie

Utilizarea **Deep Learning** a produs o crestere spectaculoasa a preciziei. Pentru Task 1, am obtinut un scor de 99.6% **Average Precision (AP)**, iar pentru Task 2, am obtinut un **mAP** de 98.6%.



Rezultatul pentru Task 1 - Bonus



Rezultatele pentru Task 2 – Bonus

5. Rezultate

Lista de mai jos rezuma performantele obtinute de cele doua abordari implementate:

- Task 1 folosind **HOG + SVM**: 65.1% AP
- Task 1 folosind **YOLOv8**: 99.6% AP
- Task 2 folosind **HOG + SVM**: 37.8% AP daphne, 27.7% AP fred, 26.5% AP shaggy, 71.7% AP velma => 40.92% mAP
- Task 2 folosind **YOLOv8**: 97.9% AP daphne, 100% AP fred, 98.9% AP shaggy, 97.7% AP velma => 98.6% mAP