Andronic Smaranda 461

**Proiect 2 CAVA**

**Task 1**

Înainte de a începe rezolvarea primului task mi-am realizat mai întâi setul de date pentru antrenare. Astfel, am comasat toate exemplele pozitive într-un folder unic, care conține poze cu fețele decupate ale personajelor, în funcție de adnotările corespunzătoare fiecărei imagini. Acest lucru l-am realizat iterând prin fișierele cu adnotări și atâta timp cât numele fișierului corespundea cu imaginea curentă la care ne aflam, atunci continuam decuparea din poza curentă.

**A screen shot of a computer program

Description automatically generated**

După care, adăugăm în folderul aferent exemplelor pozitive imaginile noastre decupate.

**A computer code on a black background

Description automatically generated**

Procedăm similar cu crearea exemplelor negative pe care le obținăm decupând aleator 3 patch-uri din imaginile inițiale pozitive, dar fără a permite intersectarea cu fețe mai mult de 0.015 în urma raportului IOU și având grijă la toate fețele existente in imagine.

A screen shot of a computer program

Description automatically generated

Totodată, în funcția care obține descriptorii pozitivi am redimensionat imaginile pozitive astfel încât să fie toate la aceeași dimensiune, și anume **self.params.dim\_window**. Am procedat la fel și pentru exemplele negative.

A screen shot of a computer program

Description automatically generated

În funcția **run(),** pentru fiecare imagine pe care vrem să detectăm personaje, aplicăm mai mulți factori de scalare, pentru a o redimensiona și a detecta, cu aceeași fereastră, fețe de dimensiuni diferite. Astfel, am implementat multi-scale sliding window, redimensionând imaginea de input. Dacă am găsit într-o anume imagine o față, pentru a-i găsi coordonatele corespunzătoare din imaginea inițială trebuie să aplicăm invers același factor de scalare aplicat și imaginii.

A screen shot of a computer program

Description automatically generated

Parametri folosiți pentru rezolvarea acestui task sunt următorii :

A screenshot of a computer code

Description automatically generated

Totodată, am integrat si detectorul YOLO, pornind de la modelul pretrained “yolo8n.pt” ( versiunea nano a modelul pentru a facilitata antrenarea rapidă ) , pe care l-am antrenat cu toate exemplele pozitive, respectând o ierarhie de fișiere strictă, anume :

datasets unde numele fisierelor din images si label corespund

└── dataset în întregime pentru același folder (fișierele din

├── images images/train au acelși nume cu cele din label/train)

│ ├── train

│ └── val

└── label

├── train

└── val

În plus, avem nevoie și de fișierul de configurare (.yaml), în care am precizat locația folderelor, precum și numărul de clase.

A computer screen with text

Description automatically generated

Înainte de a copia label-urile aferente din imaginile pozitive și de validare, datele au trebuit mai întâi normalizate în intervalul [0, 1].

A screen shot of a computer

Description automatically generated

Pentru acești parametri :  
A screenshot of a computer

Description automatically generated

Am obținut următoarele rezultate :

A graph of a graph of a number of data

Description automatically generated with medium confidence

Din ultimul grafic de mai sus, se poate deduce AP (average precision) pentru un IOU variind între 0.5 – 0.95, cu pași de 0.05. Deci pentru cazul nostru în care IOU trebuie să fie minim 0.3, modelul se descurcă cu o acuratețe de aproape 100%.

A screen shot of a computer program

Description automatically generated

Fișierele npy corespunzătoare detecției prin YOLO se vor afla tot în folderul de output evaluare/fisiere\_solutie/461\_Andronic\_Smaranda/task1 și vor avea prefixul YOLO : **“YOLO\_file\_names\_all\_faces.npy”**