Năstase Matei-Dorian
Grupa 333

Documentație Proiect 2
Concepte și Aplicații în Vederea Artificială

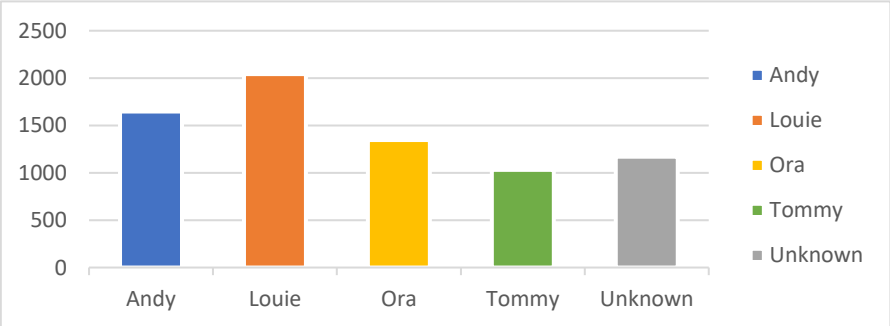
Lucrarea de față detaliază modul de abordare a problemei și pașii importanți din implementarea soluției. Cerința proiectului este compusă dintr-un task de detecție facială și unul de recunoaștere facială, aplicate pe scene din animația “Viața cu Louie”. Soluția propusă rezolvă ambele task-uri folosind paradigma sliding window.

Setul de date

Pentru rezolvarea cerințelor s-au pus la dispoziție 4000 de poze de antrenare (1000 de poze pentru fiecare personaj) și 200 de imagini de validare. Aceste imagini reprezintă scene din serial, iar fiecare scenă are adnotări corespunzătoare, ce conțin coordonatele chenarului ce încadrează o față și numele personajului din acel chenar.

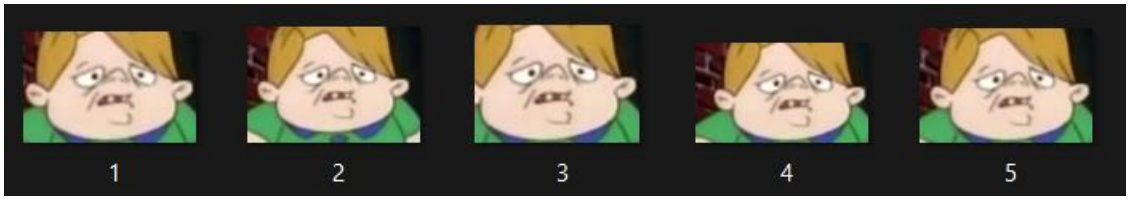
Pentru a începe antrenarea unui model de machine learning capabil de a detecta/recunoaște fețe, este necesară crearea propriului set de date ce conține exemple positive (fețe de personaje) și exemple negative (non-fețe). În fișierul *prepare dataset 2.0.ipynb* se găsesc funcțiile folosite pentru citirea datelor inițiale și extragerea datelor ce vor fi folosite la antrenare.

Generarea de exemple pozitive este realizată de funcția *extract_faces_from_annotations* care la început citea toate adnotările și tăia din fiecare imagine fețele personajelor pe baza coordonatelor citite și salva aceste imagini în directorul corespunzător (dataset/train/positives/character_name). În total s-au obținut 7236 fețe distribuite astfel:



Deoarece precizia clasificatoarelor antrenate pe aceste exemple nu era satisfăcătoare, au fost folosite mai multe metode de augumentare a datelor. S-a observat ca deși clasificatorul avea acuratețe de 100% pe fețele extrase din datele de validare, atunci când foloseam sliding window pe imaginile întregi multe fețe nu erau detectate. Acest lucru poate provine din faptul că modelul nostru este antrenat pe fețe perfect decupate, dar atunci când glisăm fereastraeste foarte puțin probabil să nimerim un astfel de exemplu. Astfel, funcția *extract_faces_from_annotations* a fost modificată pentru a adăuga pe lângă fețele din adnotări și fețele cu următoarele modificări:

- fața bordată cu toate marginile de 6 pixeli (mimează o fereastră încadrată bine dar la un aspect ratio nepotrivit)
- fața bordată cu o singură margine de 12 pixeli (aspect ratio bun, încadrare nepotriviă)
- fața din care am tăiat o margine de 12 pixeli (aspect ratio bun, încadrare nepotrivită)
- fața la care am adăugat 2 margini de 6 pixeli la dreapta și jos, sau la stânga și sus (aspect ratio bun, încadrare nepotriviă)



Aplicând aceste tehnici am reușit să mărim setul de date de 5 ori. Mai mult, înainte de a antrena SVM-uirle, pentru a avea cât mai multă diversitate în date și a evita overfitul am aplicat operațiile de flip și rotate la stânga și la dreapta pe câte o pătrime din pozele din setul de date generat.

Pentru exemplele negative am folosit funcția *extract_negative_examples*. Ideea este de generare a unor patchuri aleatore din imaginea inițială și salvarea acestora în folderul *dataset/train/negatives/normal* dacă acestea nu conțin fețe. Pentru a determina dacă un patch

conține sau nu o față s-a folosit criteriul *Intersection Over Union* împreună cu adnotările. Acest criteriu este folosit deoarece dorim ca în exemplele negative să avem și bucăți dintr-o față (ochi, păr, urechi) pentru a avea un clasificator cât mai puternic. Inițial au fost extrase patch-uri în mod aleator de dimensiuea ferestrei pe care o glisam (24x24), însă s-au remarcat următoarele probleme:

- dimensiunea patch-urilor era prea mică, multe din acestea erau doar o simplă culoare și nu conțineau informații sau pattern-uri pe care le-ar putea învăța modelul
- se generau numere mult prea apropiate, multe din patch-urile extrase semănau între ele

Pentru a rezolva aceste probleme am mărit dimensiunea patch-ului la 64x64 de pixeli și am generat numerele folosint funcția *distant_rand* care se asigură ca distanța dintre două patchuri consecutive să fie ≥ lungimea patch-ului / 2. S-au extras 100 de mii de astfel de exemple care au fost folosite în antrenarea SVM-urilor.

Extragerea feature-urilor

Odată extrase exemplele de antrenare, acestea trebuie convertite în vectori de features pentru a putea începe procesul de antrenare. Deoarece fețele personajelor sunt foarte variate (uneori personajele poartă pălării, ochelari, avem personaje de culoare) niște descriptori bazați pe culoare nu se pliază foarte bine pe această problemă, așa că s-a optat spre folosirea descriptorilor HOG care sunt invarianți la schimbările de culoare și folosesc gradienții din imagine pentru a reprezenta forma unei fețe. Funcțiile ce extrag descriptorii din imagini se găsesc în fișierul trainer 2.0.ipynb

Inițial am folosit o fereastră de 24x24 cu 9 pixeli într-o celulă hog (3x3), 4 celule într-un bloc și 9 orientări. Ulterior s-a observat că modelul obține o acuratețe mai bună folosind mai multe

orinetări, însă timpul de antrenare al SVM-urilor creștea destul de mult datorită creșterii numărului de features, așa că am crescut numărul de pixeli dintr-o celulă. Soluția finală folosește o fereastră de 36x36 de pixeli cu 6x6 pixeli pe celulă, 2x2 celule pe bloc și 18 orientări.

Este important de menționat că dimensiunea ferestrei a fost aleasă ca acesta să se împartă exact la numărul de celule și de blocuri, dar și prin analiza fețelor din setul de date: cele mai mici fațe aveau erau intre 20x20 și 30x30. S-a încercat și folosirea unor ferestre dreptunghiulare, cu dimensiuni ce păstrează proporțiile personajelor principale (andy: 30x42, ora: 30x36, louie: 36x30, tommy: 32x36) însă programul scotea mult prea multe detecții.

După ce s-au calculat descriptorii, aceștia au fost salvați în directorul *dataset/descriptors*, pentru a nu fi recalculați la fiecare rulare.

Antrenarea modelelor

Pentru task-ul 1 s-a folosit un SVM liniar, iar pentru task-ul 2 s-a folosit câte un SVM liniar pentru fiecare personaj. Antrenarea se face prin funcția *train_classifier* din fișierul *trainer 2.0.ipynb*. La antrenare s-au încercat mai multe combinații de hiperparametrii (constante diferite, creșterea tol la 1e-5), însă cele mai bune rezultate s-au obținut pentru parametrii default.

Pentru task-ul 1 pe langă cele aproape 50 de mii de exemple positive și 100 de mii de exemple negative, s-au extras încă 20 de mii de exemple puternic negative după rularea programului. Aceste exemple puternic negative au condus la o creștere cu 3.5% a MAP-ului.

Pentru task-ul 2 am folosit aceleași date ca pentru task-ul 1 doar că am schimbat etichetele (clasificatorul pentru Andy avea pozele cu Andy etichetate cu 1, iar restul personajelor și exemplele negative erau etichetate cu 0). Deoarece numărul de exemple pozitive a scăzut (pentru Tommy avem doar (5 mii de fețe), în SVM vom pune atât pozele inițiale cât și poze flipped.

Acuratețea pentru SVM-ul ce clasifică fețe și non fețe (task 1): 97.2%

Acuratețea pentru SVM-ul care clasifică:

Andy/non-Andy: 77.3%

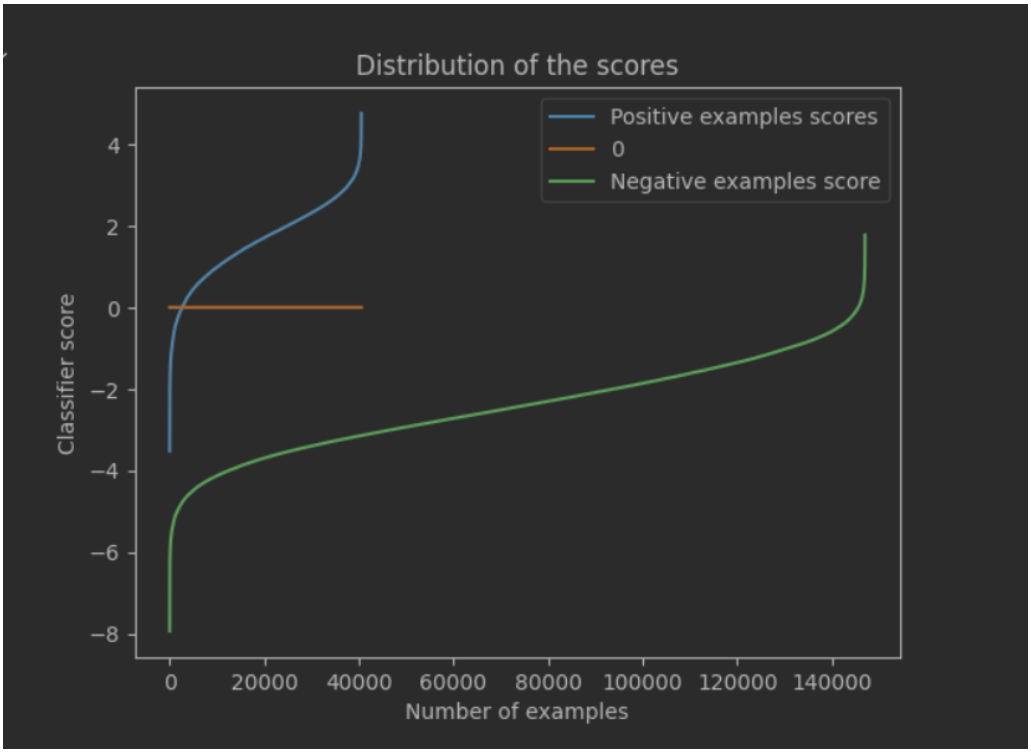
Ora/non-Ora: 76.4%

Louie/non-Louie: 78.03%

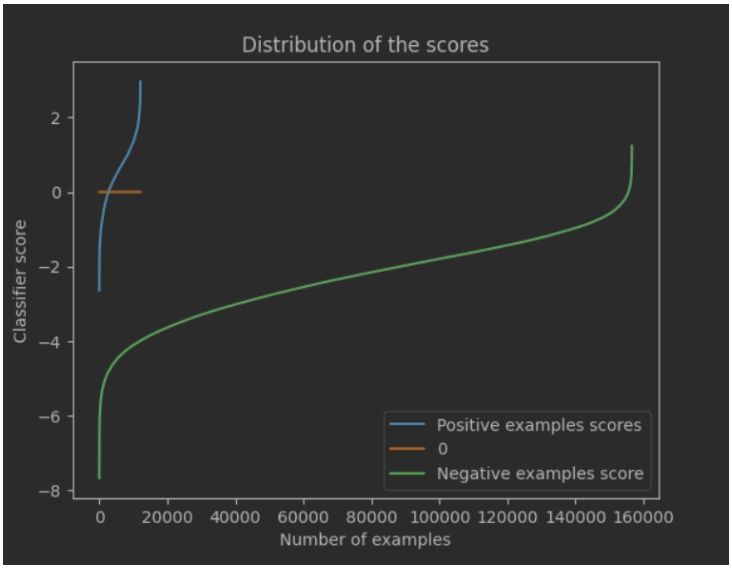
Tommy/non-Tommy: 74.3%

Năstase Matei-Dorian
Grupa 333

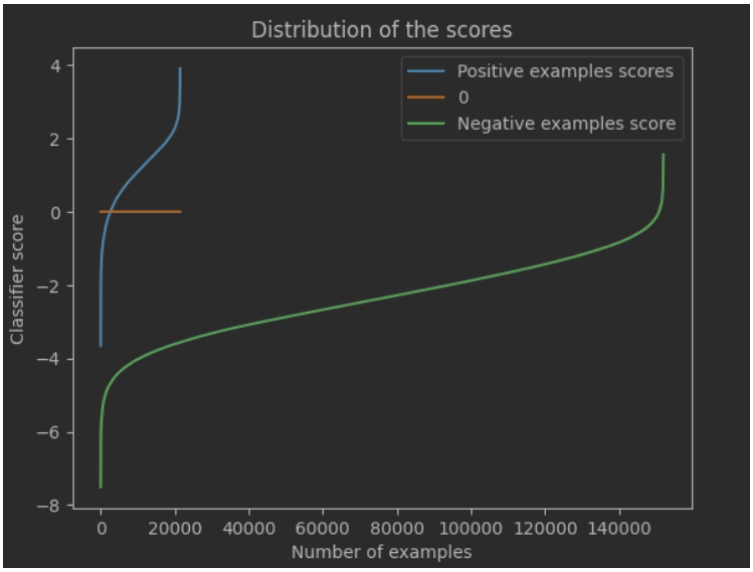
Fețe/non-fețe:



Tommy:

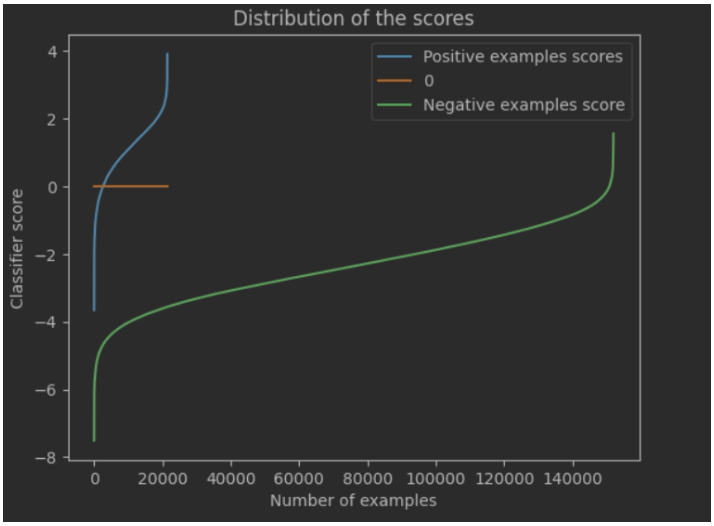


Louie:

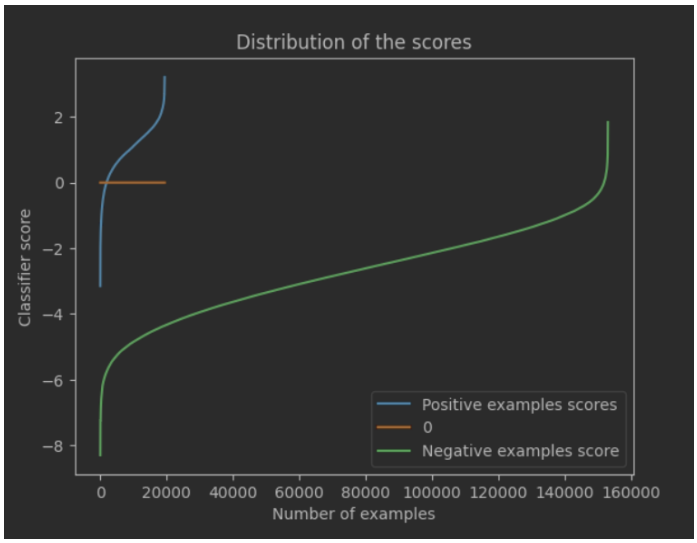


Năstase Matei-Dorian
Grupa 333

Ora:



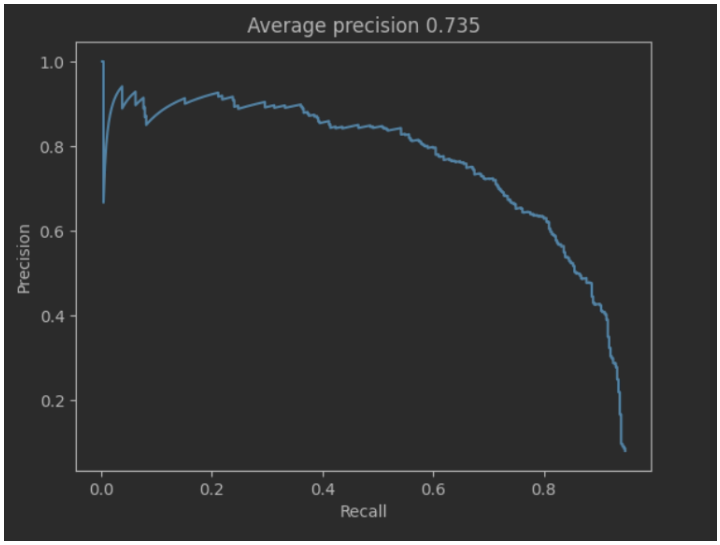
Andy:



Glisarea ferestrei:

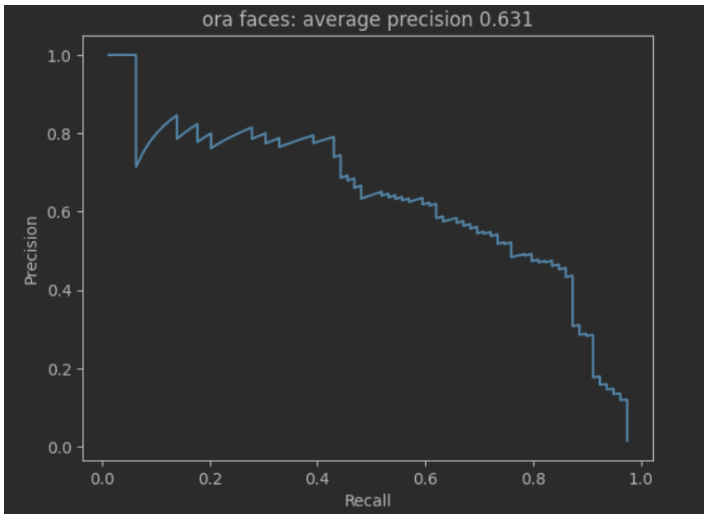
În fișierul *detector 2.0.ipynb* se găsește funcția *run* care rezolvă ambele task-uri simultan. Am optat pentru forma multi-scale a sliding window-ului, și deoarece fețele personajelor nu sunt pătrate perfecte, am ales să scalez imaginea orignială cu fx, fy diferite. Pentru fiecare poză se pornește de la α=1.1 și β=1.3 și se face down-scale cu un step de 2% la fiecare iterație. La fiecare pas se calculează descriptorii hog pentru două poze, imaginea redimensionată cu fx= α și fy= β și imaginea redimensionată cu fx= β și fy= α, iar apoi se glisează fereastra pe aceste imagini. Dacă o fereastră obține un scor > 0, aceasta este privită ca o detecție și este memorată în vectorul soluție. Înainte de returnarea rezultatelor, se aplică funcția de *Non Maximal Supression* pentru a reduce numărul de detecții. S-au obținut următoarele rezultate pentru setul de validare:

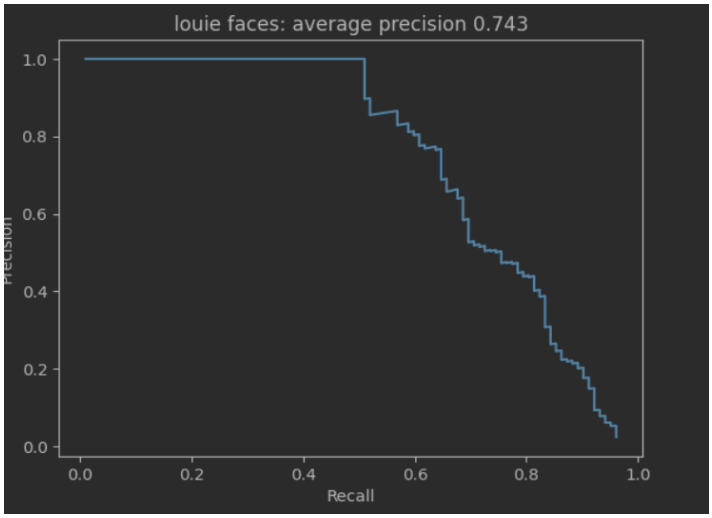
Task 1:

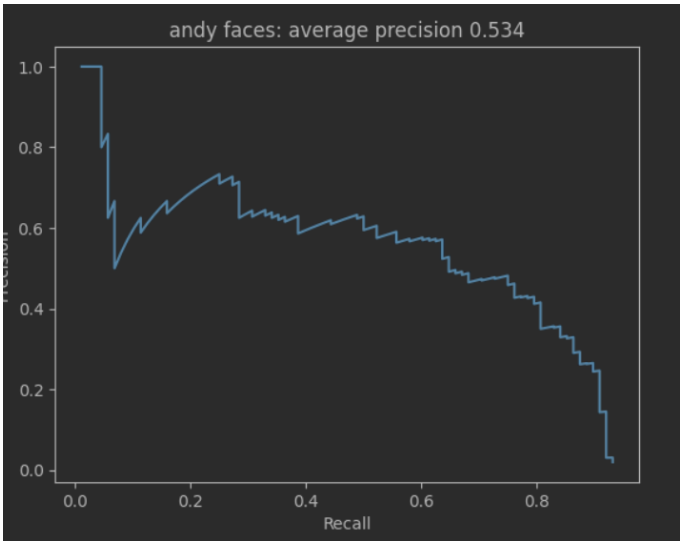


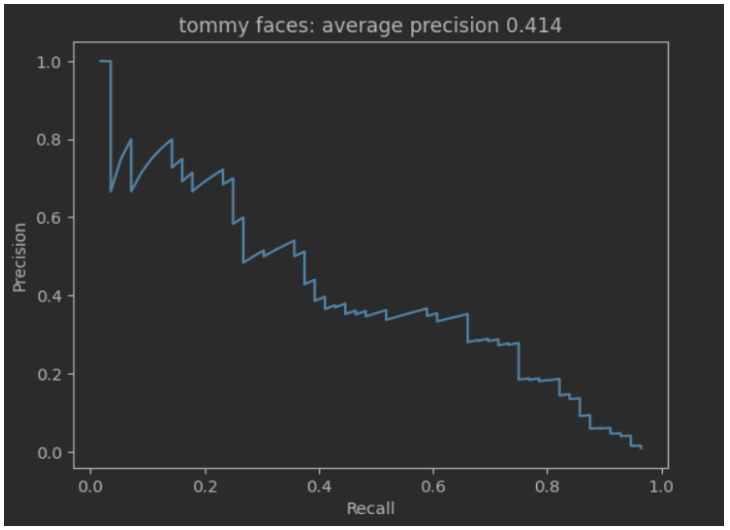
Năstase Matei-Dorian
Grupa 333

Task 2:

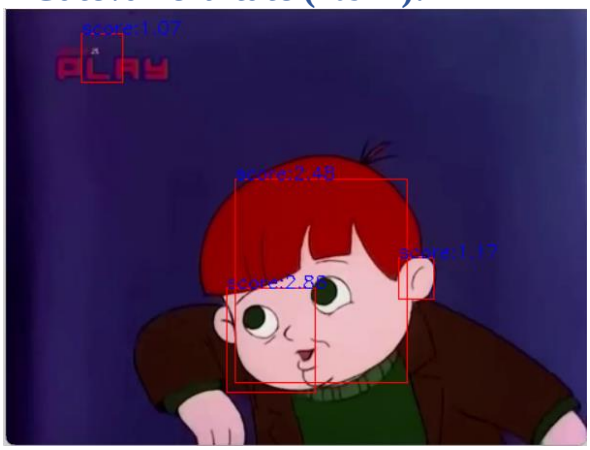


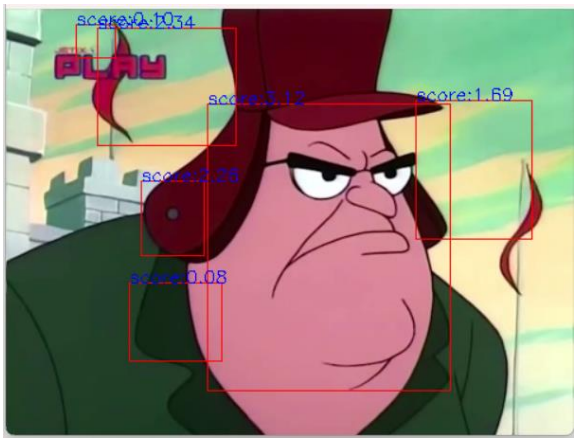




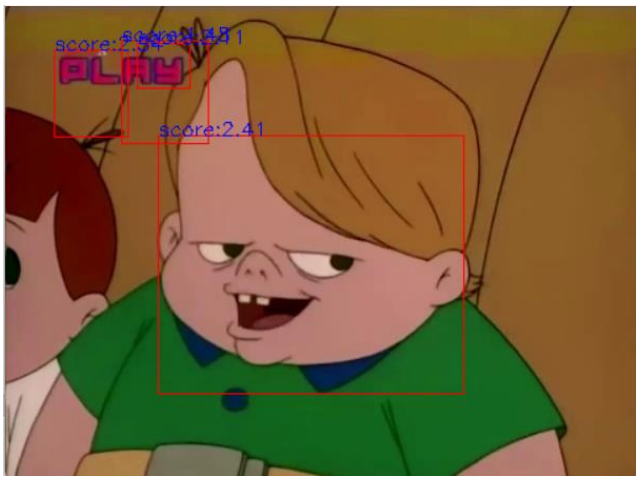


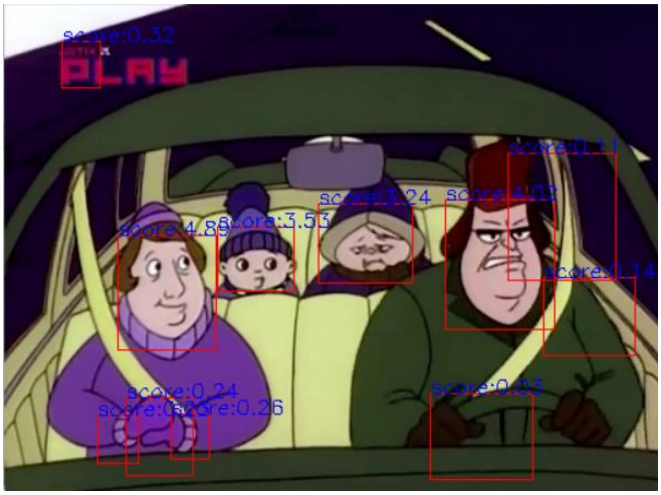
Câteva rezultate (Task 1):





Năstase Matei-Dorian
Grupa 333





Bibliografie:

- materialele de curs și laborator
- https://towardsdatascience.com/hog-histogram-of-oriented-gradients-67ecd887675f
- https://learnopencv.com/histogram-of-oriented-gradients/
- https://www.v7labs.com/blog/mean-average-precision
- https://stackoverflow.com/questions/37241930/generate-a-random-number-which-is-far-enough-from-another-number