Detectarea și recunoașterea facială a personajelor din serialul de desene animate Laboratorul lui Dexter

1. Descrierea soluției

Pentru rezolvarea temei am ales să folosesc mai multe rețele neurale convoluționale. Sistemul este bazat pe paradigma ferestrei glisante și utilizează 6 modele pentru detectarea și clasificarea fețelor din imagini. Rețelele sunt de 5 tipuri, câte o rețea ce detectează prezența în imagine a fiecărui personaj (Dexter, DeeDee, Mom, Dad) și un ansamblu format din 2 rețele ce este folosit pentru detectarea tuturor fețelor din imagine. Fiecare tip de model folosește o fereastra glisantă de marimi diferite, pentru a încadra cât mai bine tipul de față pentru care a fost antrenată. Primul pas în găsirea fețelor în imagine este parcurgerea acesteia folosind modele specifice fiecărui personaj. Acest lucru se face utilizand diferite dimensiuni ale imaginii, pornim de la 10% din dimensiunea inițială până la 250% din aceasta, adăugând 10% la fiecare pas. Fiecare fereastră este trecută prin modelul de clasificare, pentru a obține scorul acesteia, dacă scorul depășește un anumit prag, considerăm că fereastra este o detecție. După ce modelul a parcurs întreaga imagine, detecțiile acestuia vor fi introduse într-un algoritm de eliminare a detecțiilor non-maximale, pentru a păstra doar detecțiile cu cel mai bun scor. Acest procedeu este executat pentru toate tipurile de modele, singura diferență este pentru modelul ce detectează toate fețele din imagine. După ce acest model parcurge întreaga imagine, concatenăm detecțiile maximale, găsite de rețelele specializate pentru fiecare personaj. Apoi este aplicat același algoritm de eliminare a detecțiilor non-maximale. Acest artificiu ne oferă o performanță mult mai bună pentru găsirea personajelor principale în poze.

2. Generare imaginilor de antrenare

Dimensiunea ferestrei glisante pentru fiecare tip de model reprezintă media tuturor exemplelor adnotate în imaginile de antrenare, pentru personajul respectiv. Generarea exemplelor pozitive a fost efectuată utilizând adnotările prezente în setul de antrenare. Acestea au fost redimensionate pentru a avea aceeași dimensiune ca fereastra asociată modelului. Pentru exemplele negative, generez 2000 sau 3000 de imagini pentru fiecare coeficient, începând de la 50% din imaginea inițială, până la 250% din aceasta, rezultând în 42000 sau 63000 de exemple negative. Pentru a genera aceste exemple, iterez prin imaginile de antrenare până când ajung la numărul dorit. In fiecare imagine, aleg aleator o fereastră, iar apoi verific să aibă un scor de IoU(Intersection over union) mai mic de 0.5 cu adnotările personajului de care aparține fereastra. Spre exemplu dacă generăm datele de antrenare pentru Dexter, vom compara potențialul exemplu cu adnotarile lui Dexter, prezente în imagine.

3. Arhitectura rețelelor neurale convoluționale

Rețelele neurale convoluționale au o arhitectură simplă, fiind formate din doar 5 straturi convoluționale, cu filtre de dimnesiune 3x3, iar numărul acestora crește de la 32 până la 512, dublând la fiecare pas numărul lor. După fiecare dintre acestea urmează un strat de normalizare, un start de activare, ce utilizează funcția ReLU și un start de pooling maximal. În locul ultimului strat de pooling maximal folosesc unul de adaptive average pooling, pentru a reduce dimensiunea la un vector de 512 elemente. Dupa acesta, urmează un strat dense și un strat de activare ce utilizează funcția de activare sigmoid pentru a ajunge la probabilitatea de existența a personajului sau a unei fețe.

În urma unor experimente, am descoperit faptul că pot obține rezultate semnificativ mai bune pentru modelele Dad și Mom dacă utilizez imagini în formatul gray scale. Din acest motiv, respectivele modele primesc imagini ce au un singur canal.

```
        nn.Conv2d(1, 32, (3, 3)),
        nn.BatchNorm2d(32),
        nn.ReLU(),
        nn.MaxPool2d(kernel_size=2, stride= 2),
        nn.Conv2d(32, 64, (3, 3)),
        nn.BatchNorm2d(64),
        nn.ReLU(),
        nn.MaxPool2d(kernel_size=2, stride= 2),
        nn.Conv2d(64, 128, (3, 3)),
        nn.BatchNorm2d(128),
        nn.ReLU(),
        nn.MaxPool2d(kernel_size=2, stride= 2),
        nn.Conv2d(128, 256, (3, 3)),
        nn.BatchNorm2d(256),
        nn.ReLU(),
        nn.MaxPool2d(kernel_size=2, stride= 2),
        nn.Conv2d(256, 512, (3, 3)),
        nn.BatchNorm2d(512),
        nn.ReLU(),
        nn.AdaptiveAvgPool2d((1, 1)),
        nn.Flatten(),
        nn.Linear(512, 1),
        nn.Sigmoid()
```

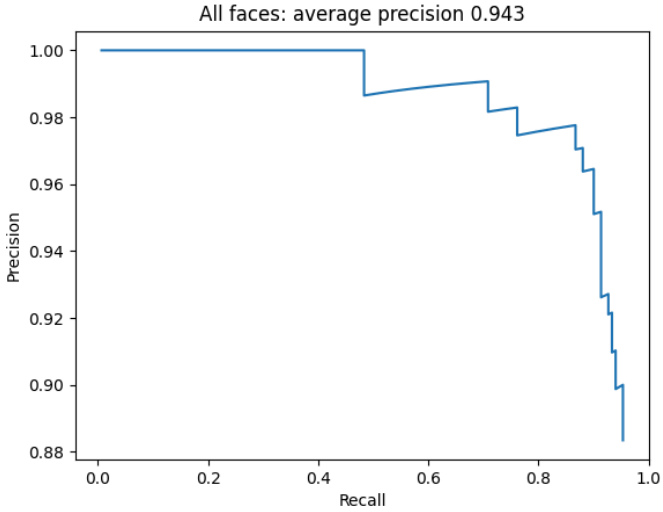
4. Antrenarea modelelor

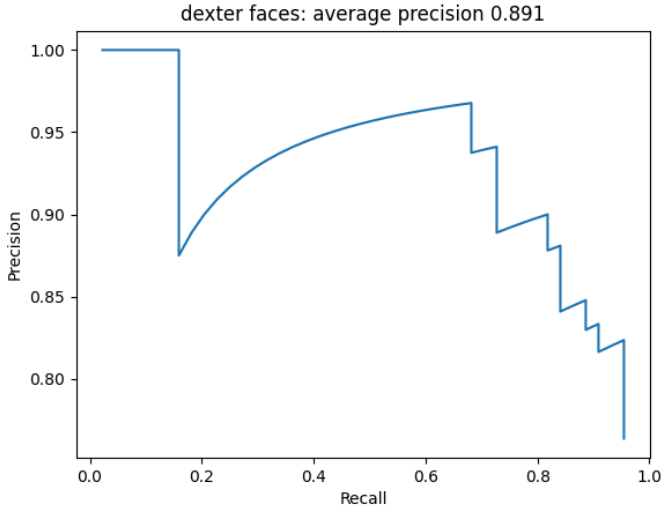
Pentru antrenarea modelelor am utilizat funcția de pierdere Binary Cross Entropy, specifică problemelor de clasificare binară. Ca algoritm de optimizare am ales Adam cu o rată de de învățare de 0.001 și un batch size de 64 de exemple. Numărul de epoci diferă pentru fiecare

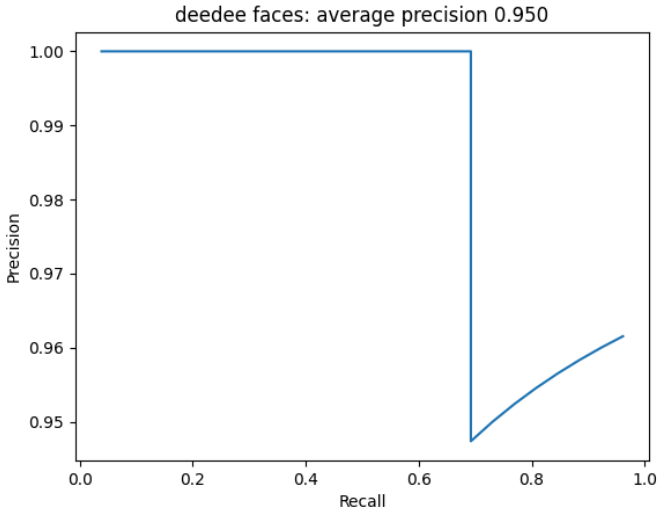
model și a fost ales prin testarea mai multor variante. Astfel modelul pentru detectarea lui Dexter este antrenat pentru 10 epoci, cel pentru Dad este antrenat pentru 14 epoci, cel pentru DeeDee este antrenat pentru 11 epoci, cel pentru Mom este antrenat pentru 20 de epoci, iar cele pentru detecția tuturor fețelor sunt antrenate pentru 14 și respectiv 16 epoci.

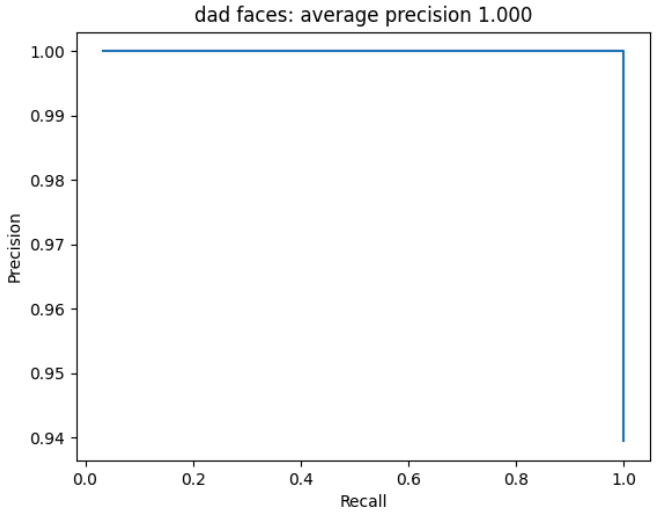
5. Rezultate

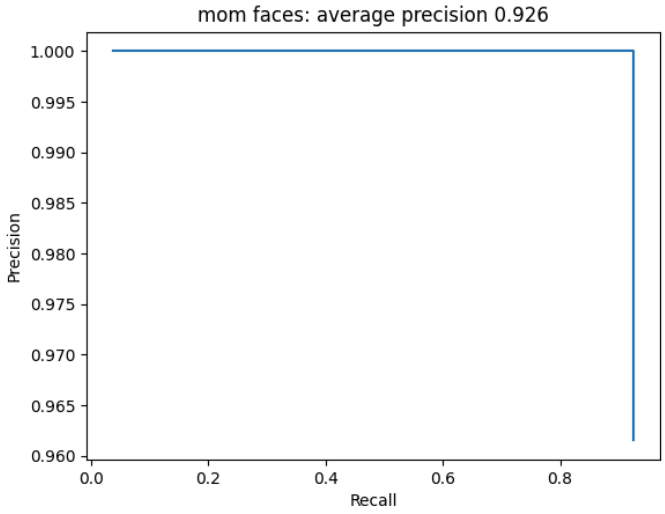
Utilizând sistemul descris anterior am reușit sa obțin rezultate, ce în opinia mea sunt foarte bune. Următoarele grafice sunt pentru primele 100 de imagini din setul de validare.










mom faces: average precision 0.926
Precision
Recall
1.000
0.995
0.990
0.985
0.980
0.975
0.970
0.965
0.960
0.0
0.2
0.4
0.6
0.8