Detectarea și recunoașterea facială a personajelor din serialul de desene animate Familia Flinstone

I. Descrierea proiectului

Proiectul constă in implementarea unei soluții care detectează fetele personajelor aflate în fiecare imagine în parte, precum și clasificarea acestora în 4 clase: Fred, Barney, Wilma, Betty.

II. Setul de date

Setul de date include poze RGB din serial, structurate in folderele antrenare și validare.

III. Rezolvare

Pentru rezolvarea primei cerințe, care face referire la detectarea faciala a personajelor am folosit o arhitectura de rețea neuronală convolutională(CNN) cu următoarea configurație.Modelul de clasificare a fețelor este construit folosind API-ul Keras din TensorFlow și este compus dintr-o secvență de straturi convoluționale, urmate de straturi de pooling și de straturi dense, inclusiv dropout pentru regularizare.

- **Primul strat convoluțional (**conv1**)**:
  - Acest strat aplică 32 de filtre convoluționale de dimensiunea 3x3 peste imaginea de intrare.
  - Funcția de activare 'relu' este folosită pentru a introduce non-linearitatea.
- **Primul strat de pooling (**maxpool1**)**:
  - Reduce dimensiunea caracteristicilor învățate prin aplicarea max pooling cu un kernel de 2x2.

- **Al doilea strat convoluțional (**conv2**)**:
  - Extinde reprezentarea caracteristicilor prin aplicarea a 64 de filtre convoluționale de 3x3.
  - Urmărește același strat de activare 'relu'.
- **Al doilea strat de pooling (**maxpool2**)**:
  - Continuă să comprime spațiul caracteristicilor, reducând dimensiunea în jumătate.
- **Al treilea strat convoluțional (**conv3**)**:
  - Profundizează învățarea caracteristicilor cu 128 de filtre convoluționale de 3x3.
  - Funcția 'relu' rămâne funcție de activare.
- **Al treilea strat de pooling (**maxpool3**)**:
  - Aduce o compresie și mai mare a reprezentării caracteristicilor.

După straturile convoluționale și de pooling, urmează o secvență de straturi pentru clasificare:

- **Flatten (**flatten**)**:
  - Transformă matricea 3D rezultată din ultimul strat de pooling într-un vector 1D pentru a putea fi procesată de straturile dense.
- **Primul strat dens (**dense1**)**:
  - Un strat dens cu 128 de neuroni și activarea 'relu' care primește vectorul aplatizat.
- **Dropout (**dropout**)**:
  - Regularizează modelul prin 'închiderea' aleatorie a unor conexiuni între neuroni cu o probabilitate de 0.5, pentru a preveni supraînvățarea.
- **Al doilea strat dens (**dense2**)**:
  - Stratul de ieșire cu un singur neuron care folosește activarea 'sigmoid' pentru a produce o probabilitate, indicând clasificarea binară (de exemplu, fata vs. Non-fata).

Rețeaua funcționează pe date cu formatul înălțime = 96, lățime = 96, în format RGB. Pentru procesarea datelor m-am asigurat ca imaginile se afla în formatul (96, 96, 3), le-am transformat în numpy array, pe care ulterior l-am normalizat utilizând împărțirea la 255.

Se utilizează conceptul de variable_sliding_window pentru a căuta obiecte într-o imagine la diferite scale. Se începe cu o fereastră de dimensiune minimă de (40, 40), algoritmul glisează fereastra peste imagine cu un pas de 20 de pixeli. După fiecare ciclu complet, dimensiunea ferestrei este mărită cu un factor de 1.2 până la atingerea dimensiunii maxime de (130, 130). Această metodă asigură că obiectele de diferite dimensiuni pot fi detectate eficient.

Pentru antrenarea modelului am folosit 6000 de date ce conțin fețe și 30000 de non-fețe. Pentru a crește acuratețea modelului, am obținut elemente puternic negative. Aceste elemente puternic negative au fost obținute astfel: după ce am antrenat modelul folosind datele de antrenare, iar ulterior pe același set de date de antrenare am realizat o prezicere, iar toate datele au fost afișare în folder-ul “train”. Manual am extras fețele personajelor, acestea fiind șterse. Ulterior, am antrenat rețeaua folosit și aceste date. Pentru a obține date negative, am extras aleator patch-uri din imaginile de antrenare, patch-uri care nu se suprapun cu fețele personajelor.

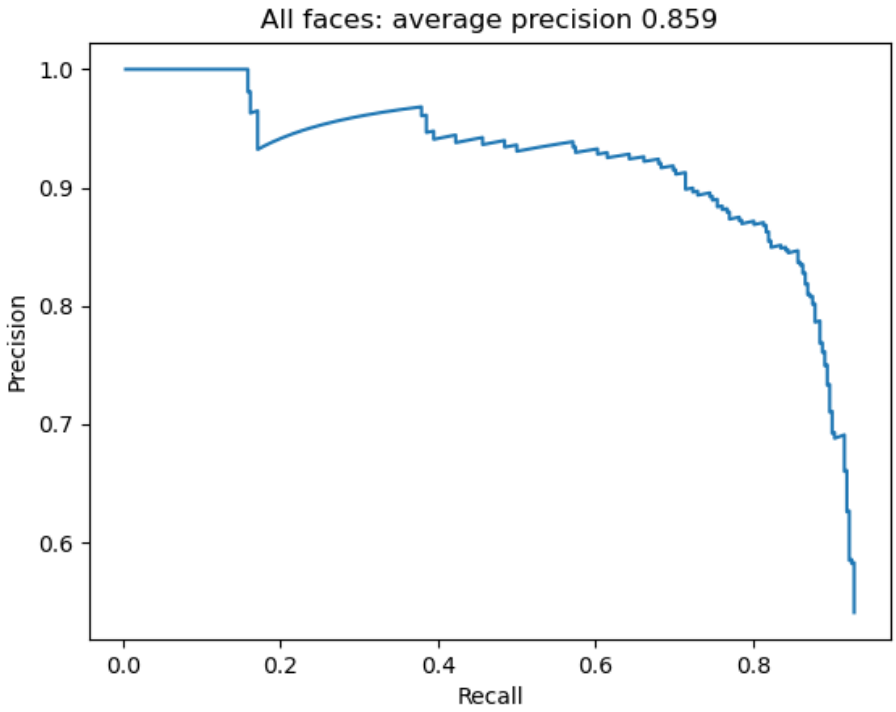
Am utilizat early_stopping(un callback din TensorFlow) care oprește antrenamentul modelului atunci când o metrică monitorizată, în cazul algoritmului meu, 'loss', nu se îmbunătățește pentru un număr de 4 epoci. Antrenamentul se va opri dacă nu există îmbunătățiri ale pierderii pentru 4 epoci consecutive. Acest mecanism ajută la prevenirea overfitting-ului.

În plus, se folosește non_maximal_suppression este utilizată pentru a reduce numărul de detectii suprapuse în procesarea imaginilor, asigurându-se că doar detectiile cele mai probabile sunt păstrate. Funcția primește ca intrare detectii ale obiectelor într-o imagine (image_detections), scorurile asociate fiecărei detectii (image_scores), și dimensiunea imaginii (image_size). Detectiile

Training history

sunt suprimate dacă se suprapun semnificativ cu o detecție cu un scor mai mare, sau dacă centrul unei detecții se află în interiorul alteia. În acest proces, se ajustează detecțiile care depășesc limitele imaginii și se aplică un prag pentru raportul de suprapunere (IoU). În final, funcția returnează detecțiile și scorurile corespunzătoare care sunt considerate maximale, adică cele mai probabil să reprezinte obiecte reale în imagine.

Creșterea acurateții modelului a fost asigurată și de funcția “is_skin_color_present”care evaluează prezența culorii pielii într-o imagine a unei fețe prin conversia acesteia în spațiul de culoare YCbCr și aplicarea unei măști pentru a detecta culoarea pielii. Se definește un interval specific de culori YCbCr reprezentativ pentru piele și se calculează procentul de pixeli care se încadrează în acest interval. Dacă mai mult de 50% din regiunea feței conține culoarea pielii, funcția returnează `True`.



Rezultat task 1

Pentru taskul 2 am folosit aceeași arhitectura ca la prima cerința. Clasificarea celor 4 personaje s-a realizat folosind 4 modele: model_fred, model_barney, model_betty, model_wilma. Pentru fiecare din parte se realizează o clasificare binară de tipul Fred - non-Fred, Barney - non-Barney, Betty - non-Betty, Wilma - non -Wilma. Pentru a creste acuratețea modelelor am furnizat un număr dublu de exemple pozitive, număr obținut prin întoarcerea pozelor ce reprezintă personajul căruia i se creează clasificatorul.

Unknown

Fred

Fred

Barney

Unknown

Fred

Wilma

Betty

Fiecare poza ce indică o față a fost trecută prin cele 4 clasificatoare. Se obțin 4 predicții(probabilități). Se alege probabilitatea cea mai mare, acesta fiind ce care indică ce personaj se afla în imaginea respectivă. În antrenarea modelor am folosit lr=0.001, optimizator Adam, batch_size = 64 și EarlyStopping, care oprește antrenamentul modelului atunci când o anumită metrică, în cazul meu - 'loss', nu se îmbunătățește pentru un număr de epoci consecutiv, în exemplul meu fiind vorba de 4 epoci.