**Identificarea hemoragiei craniene: Identificarea sangerarii in imagini CT cu creierul**

Scopul este impartirea in doua clase a imaginilor CT cu creierul: una pentru cele unde e hemoragie (eticheta 1) si una unde este normal (eticheta 0). Fiecare “sample”(imagine) este o imagine alb-negru de 224x224 pixeli.

Fiecare exemplu este assignat uneia din cele doua clase. Setul de antrenare contine 17.000 de imagini etichetate. Setul de test consista in 5.149 de exemple(imagini). Etichetele pentru setul de test nu sunt date, scopul este aflarea lor.

1. **Datele**

Arhiva cu imagini (setul de antrenare de 17.000 si setul de test de 5.149), unde imaginile au nume 000001.png, 017000.png pana la 022149.png.

Etichetele asociate setului de antrenare intr-un fisier txt: id,clasa, unde id-ul este numele imaginii si clasa 0 sau 1. De exemplu, 000001,0, ceea ce inseamna ca imaginea 000001.png apartine clasei 0, adica nu exista hemoragie.

Imaginile si etichetele pentru setul de antrenare au fost puse pe Google Drive si apoi incarcate pe Google Colab.

A fost creat un array cu etichetele extrase din fisierul txt, care ulterior a fost transformat intr-un numpy array, pentru a putea lucra cu datele.

A fost creat un array cu imaginile citite cu functia imread din OpenCV, unde path-ul a fost dat ca parametru.

A fost nevoie ca dimensiunea vectorului (numpy) de etichete sa fie reformatata, deoarece pentru model este necesar un vector 2D, adica o matrice.

Pentru model, a fost luata dimensiunea vectorului de features [1:] si a fist facuta tuplu, deoarece este immutable.

1. **Arhitectura modelului si optimizatorul**

Crearea unui model secvential (Sequential): a fost aleasa o retea neuronala convolutionala (convoluted) consistand in niste straturi convolutionale, straturi de pooling, de flattening etc.

A fost utilizat Keras, un API pentru retele neutonale, scris in Python.

Au fost adaugate:

* 4 straturi convolutionale cu 128, 64, 32, respectiv 16 filtre, cu dimensiunea kernel-ului (7, 7), (6, 6), (5, 5), respectiv (4, 4) si cu functia de activare ‘relu’, impreuna cu straturi de normalizare si de pooling pentru a reduce dimensionalitatea
* Un strat de flatten si 5 straturi de dense cu 1024, 512, 256, 128, respectiv 2 unitati (pentru ca output-ul final sa aiba dimensionalitate 2, precum numarul de clase) si cu functia de activare ‘relu’ si pentru ultimul strat functia de activare ‘softmax’



A fost ales optimizatorul de tip Adam (Adaptive Moment Estimation) cu o rata de invatare de 1e-5.

Au fost facute diverse experimente, si cu optimizator de tip SGD (Stochastic gradient descent), dar s-au obtinut rezultate mai bune cu cel de tip Adam. De asemenea, a fost variata si rata de invatare, dar rezultatele cele mai bune au fost obtinute cu o rata de 1e-5.

1. **Antrenarea modelului**

Modelul a fost antrenat pe date cu un numar de 15 epoci (cand au fost adaugate mai multe, parea sa apara overfitting) si cu un batch\_size de 64 de exemple.

1. **Prezicerea etichetelor pentru setul de test**

Pentru prezicere, a fost creat un fisier pentru a pune etichete celor 5.149 imagini din setul de test, unde au fost notate la fel ca in fisierul cu etichete pentru setul de date de antrenare.

A fost “citita” fiecare imagine din cele 5.149 (de la 17.001 pana la 22.149) si a fost aplicata functia ‘predict’ pe imaginea convertita in numpy array.

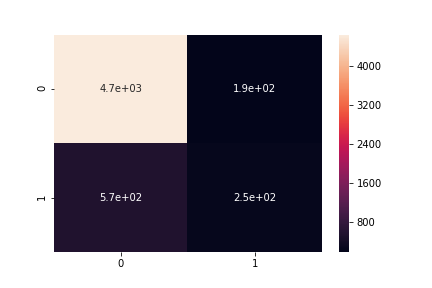
1. **3-fold cross-validation cu confusion matrix**

Prin 3-fold cross-validation, setul de date a fost impartit la intamplare in 3, modelul a fost creat si antrenat pe fiecare parte de date din cele 3, a fost evaluat prin calculul pierderii si acuratetii si apoi a fost calculate matricea de confuzie si plotata.

Prima parte de date

Pierdere: 0.41

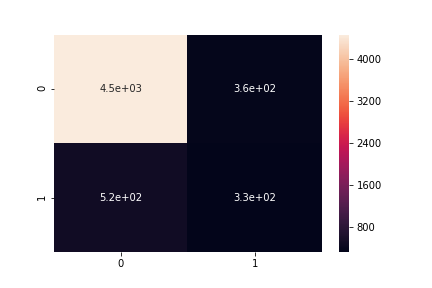
Acuratete: 0.86

****

A doua parte de date

Pierdere: 0.46

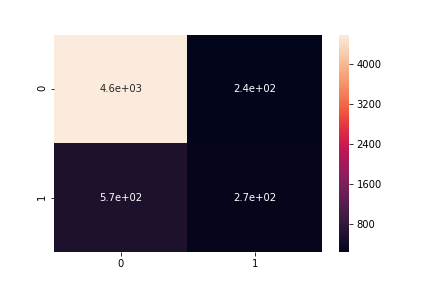
Acuratete: 0.85

****

A treia parte de date

Pierdere: 0.41

Acuratete: 0.86

****