CT Scan Classification

Documentație

Proiect Machine Learning

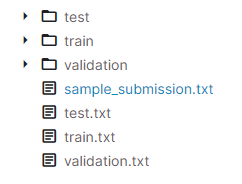
Hermeneanu Mara

Grupa 312

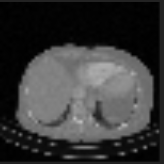
*Introducere*

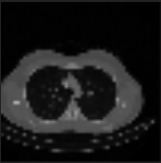
Scopul proiectului a fost acela de a antrena modele de machine learning pe un set de date constând în tomografii ale unor plamani și a clasifica imaginile CT în una din cele trei categorii: nativ (0), arterial (1), venos (2). În paginile ce urmează voi descrie abordarea folosită și tipurile de modele implementate ( MLP și CNN).

*Seturile de date*

Fișierele train.txt și validation.txt conțin pe fiecare linie numele imaginii și label-ul respeciv, în formatul: nume\_imagine, label\_imagine. Imaginile propriu-zise se regasesc în folderele train și validation.

Fișierul test.txt conține pe fiecare linie denumirea unei imagini, iar imaginile propiu-zise se găsesc în folderul test.

Imaginile sunt în fomat png, grayscale, cu o rezoluție de 50x50 pixeli.

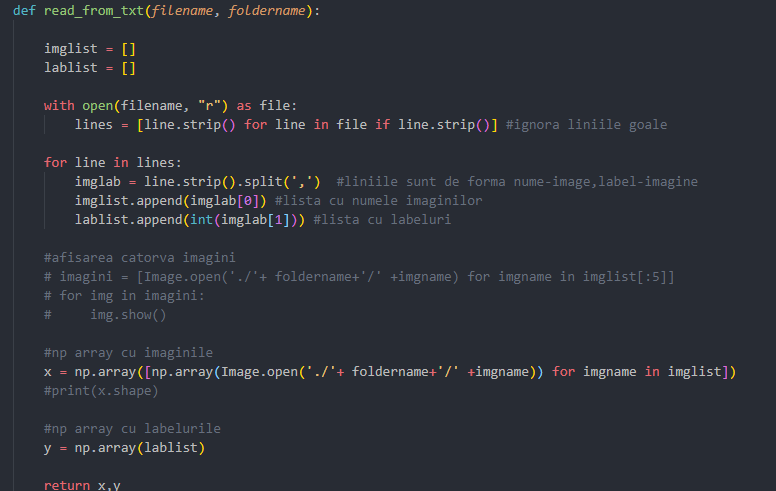


*(cateva din CT Scan-urile de train afisate in program)*

1. *Multi Layer Perceptron Classifier*

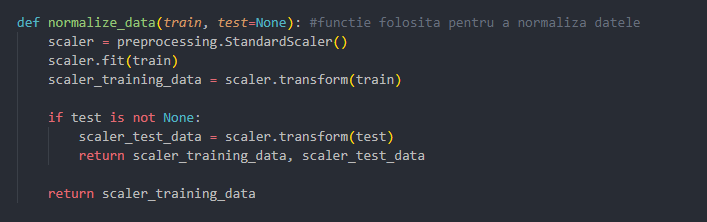
*Încărcarea seturilor de date*

Funcția **read\_from\_txt(filename, foldername)** citește din fișier numele imaginilor și label-urile corespunzatoare și le adauga în două liste separate. Având numele imaginilor și numele folderului în care se află imaginile, folosesc în continuare **Image.open()** pentru ale încărca într-un np.array, iar lista de label-uri o transform de asemenea într-un np.array. Funcția va returna np.array-ul de imagini și np-array-ul de labels.



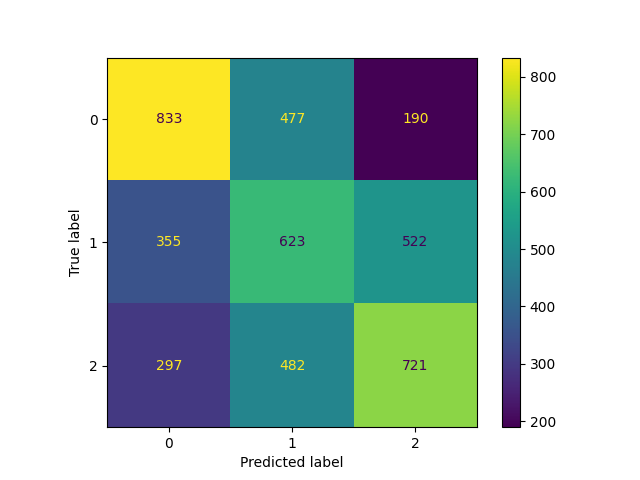
*Normalizarea datelor*

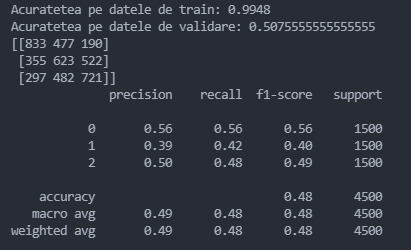
Pentru normalizarea datelor am folosit o funcție implementată la laborator **normalize\_data(train\_data, test\_data)**, care primește ca parametri datele de antrenare, respectiv de testare și folosește **preprocessing.StandardScaler()** pentru a le normaliza. Înainte de a normaliza imaginile a fost nevoie sa fac reshape pentru a reduce o dimensiue. Astfel, pentru imaginile de train am schimbat shape-ul din (15000,50,50) în (15000,2500), pentru imaginile de validation din (4500,50,50) în (4500,2500), iar pentru imaginile de test din (3900,50,50) în (3900,2500).



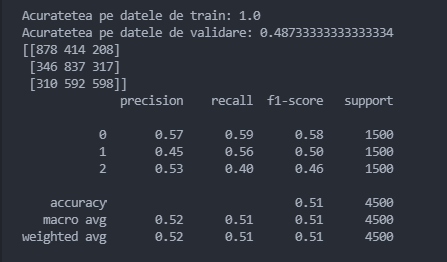
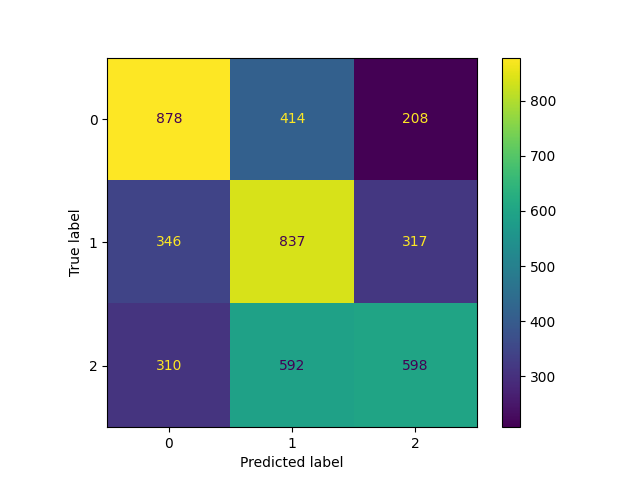
*Antrenarea modelului și hyperparameters*

Pentru alegerea parametriilor am încercat mai multe combinații posibile, cea mai reușită fiind cea cu funcția de activare „relu”, solver-ul implicit „adam” , learning\_rate „adaptive”, max\_iter = 2000 și hidden\_layer\_sizes=(200,100,50). Acuratețea maximă obținută a fost în jur de 0.52.

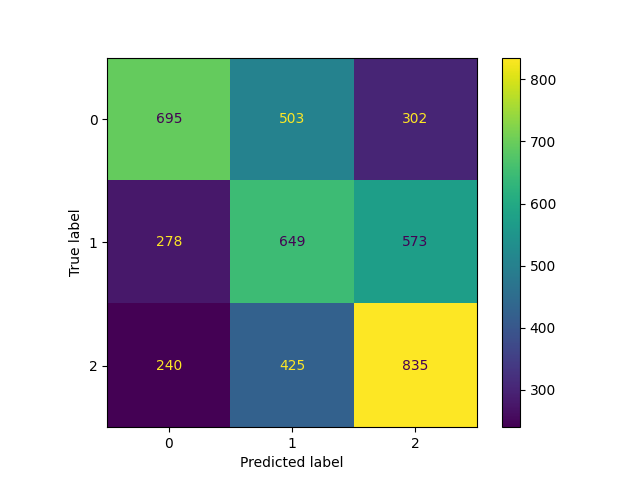
*Comparație între solverele* ***adam*** *și* ***sgd***

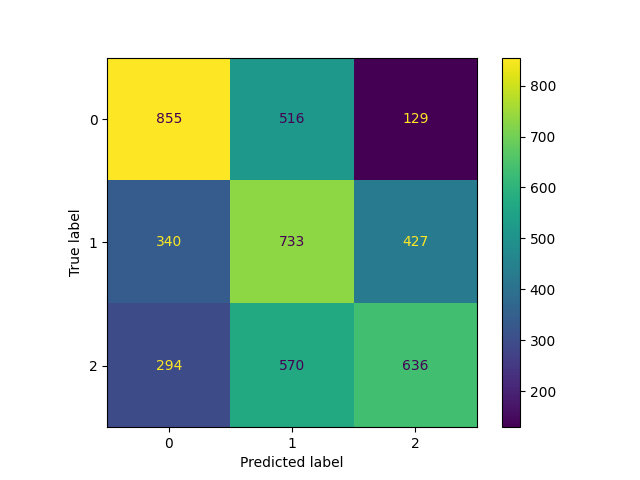
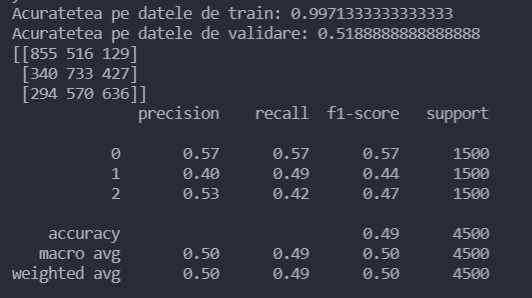
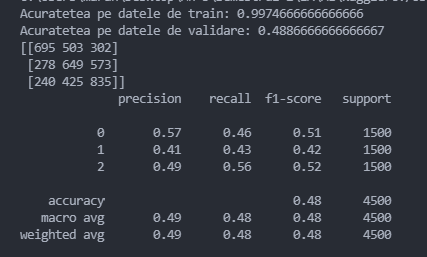
**

*(rezultate obținute cu adam, max\_iter=1000,learning\_rate=’adaptive’, activation = 'relu')*

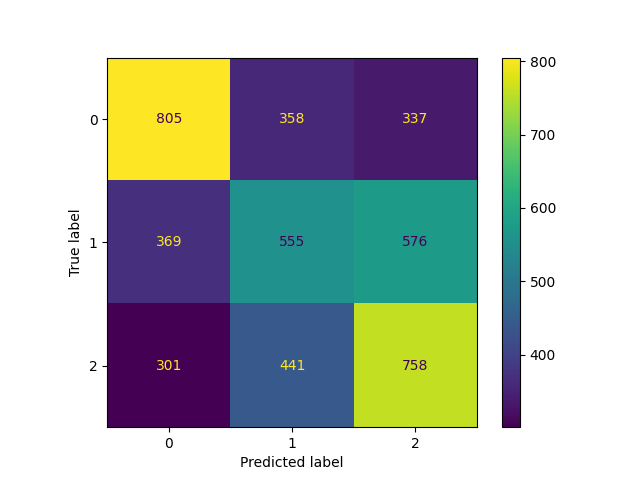
********

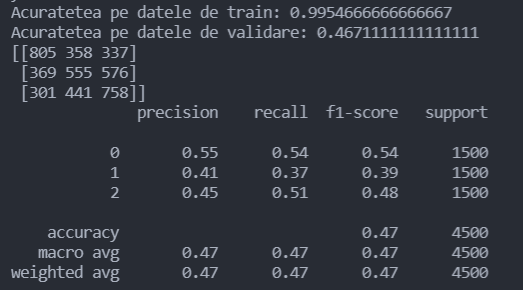
*(rezultate obținute cu sgd, max\_iter=1000,learning\_rate=’adaptive’* *activation = 'relu')*

*Comparație între learning rate* ***constant*** *și* ***adaptive/relu*** *și* ***logistic***

*******(rezultate obținute cu learning rate constant, max\_iter=2000,* *activation = 'relu',solver=’adam’)*

*(rezultate obținute cu learning rate adaptive, max\_iter=2000, activation = 'relu', solver=’adam’)*

******

******

*(rezultate obținute cu learning rate adaptive, max\_iter=2000, activation = 'logistic', solver=’adam’)*

1. *Convolutional Neural Network*

Având în vedere că acuratețea obținută cu MLP a fost una destul de nesatisfăcătoare, am decis să folosesc un algoritm bazat pe CNN, acest tip de rețea fiind extrem de populară pentru clasificare de imagini. Pentru a implementa acest tip de clasificare am folosit biblioteca PyTorch, ghidându-mă după documentația oficială.

*Încărcarea seturilor de date*

Pentru a face batch training am implementat o clasa CTScanData care moștenește clasa Dataset și conține metodele \_\_getitem\_\_ și \_\_len\_\_ necesare pentru construirea de DataLoaders pornind de la obiectele de tip CTScanData. În constructorul clasei CTScanData încarc imaginile și label-urile corespunzătoare într-un mod similar cu funcția *read\_from\_txt(filename, foldernam*e), cu deosebirea că transform lista de np.array intr-o listă de tensori (***torch\_transf.ToTensor()***) pe care îi normalizez (***torch\_transf.Normalize(media,dev)***).

*Arhitectura rețelei*

Primul model a fost conceput având 3 straturi convolutionale, 3 straturi de dropout și 3 straturi liniare cu 4320,2160 și în final 3 features. Pentru straturile convoluționale am ales sa dublez numarul de out\_channels pe fiecare strat, iar pentru kernel am ales o dimensiune constanta de 3x3. Straturile de dropout anuleaza elementele tensorului de input cu o anumita probabilitate, împiedicând totodată overfitting-ul. Pentru aceste probabilități am ales valorile 0.5,0.4 și 0.3.



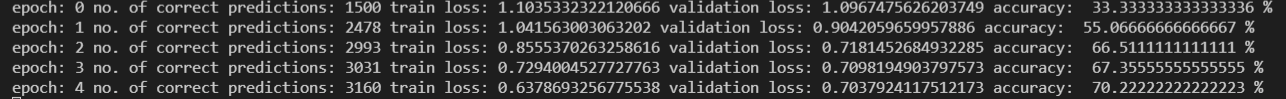
În metoda forward am ales să folosesc max pooling pentru feature extraction cu un filtru de dimensiune 2x2 după fiecare strat convoluțional, astfel că dimensiunile imaginii vor fi înjumătățite. Funcția de activare folosită a fost relu. Optimizer-ul folosit a fost adam cu un learning rate de 0.0005, iar acuratețea obținută cu acest model a fost în jur de 0.70.

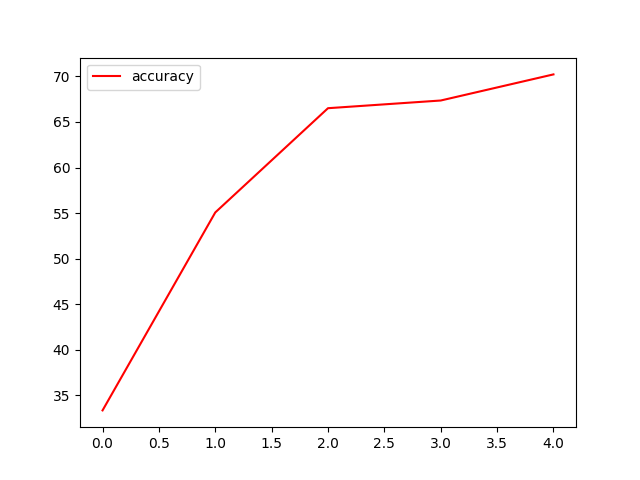
Incercând să obțin o acuratețe mai mare decât cea a primului model, am adăugat un strat convoluțional în plus și am adaugat straturile de dropout doar după straturile convoluționale 2 și 3. De asemenea, am mărit succesiv dimensiunea kernel-ului, am micșorat numărul de features pentru straturile liniare și am folosit ca ultimă funcție activatoare funcția softmax cu dim=1. În ceea ce privește optimizer-ul, am ales să rămână adam, însă am mărit learning rate-ul la 0.0009. Acuratețea obținută cu acest model a fost mai bună, oscilând între 0.72 și 0.75.

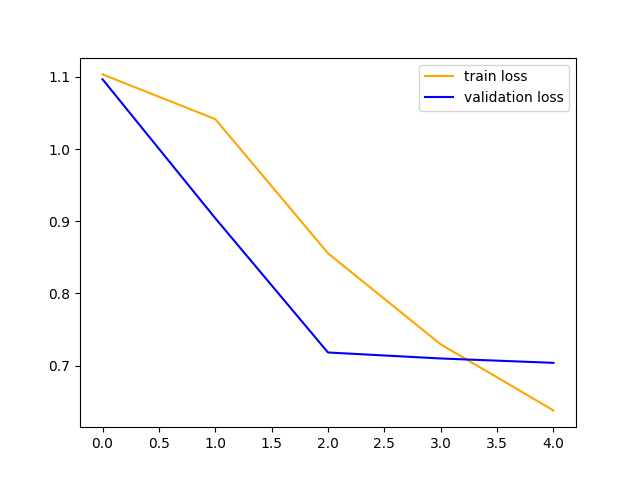


Pentru fiecare epocă am calculat training\_loss, respectiv validation\_loss pentru a vedea cum evoluează modelul (dacă există semne de undefitting sau overfitting) și am reținut numărul de imagini corect clasificate pentru a calcula acuratețea dupa formula 100 \* imagini\_corect\_clasificate/total\_imagini.

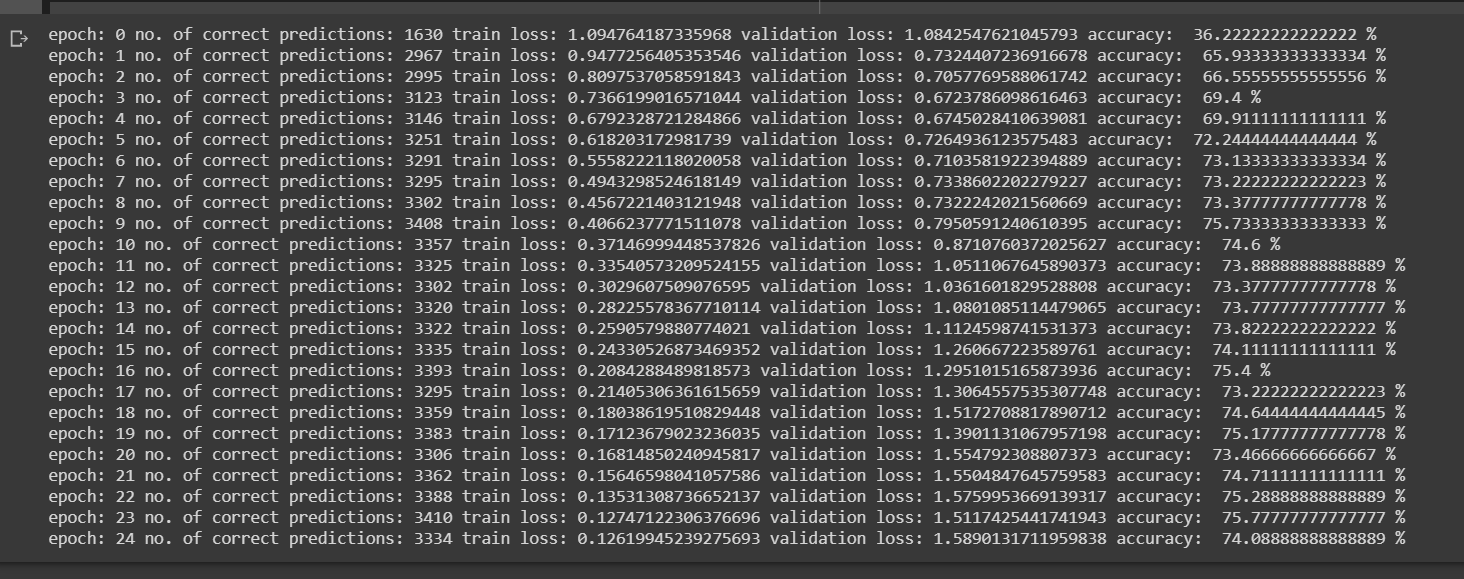
*Rezultate pentru primul model*

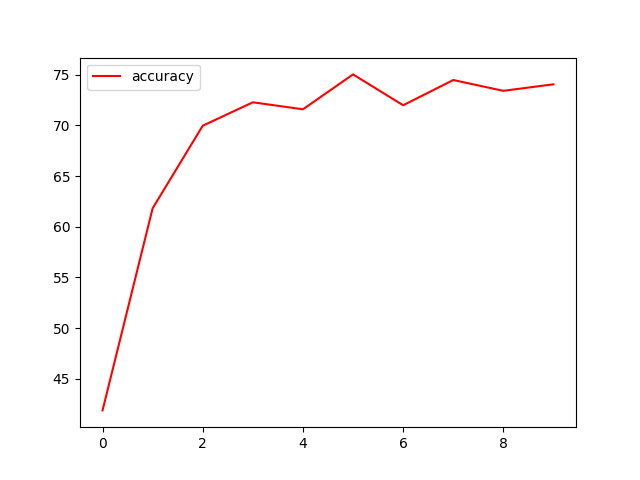
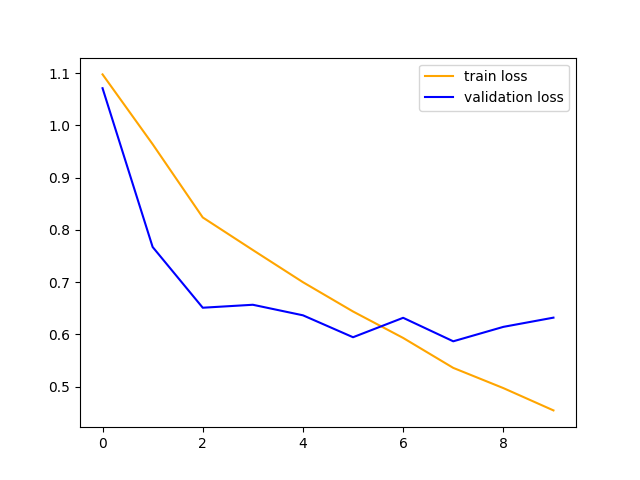
**

**

**

*(graficele pentru 5 epoci)*

*Rezultate pentru cel de-al doileal model*

**

*(graficele pentru 10 epoci)*