CT Scan Classification

Bouruc Petru-Liviu

21/05/2021

Contents

1 Introducere & Descrierea proiectului

Acest proiect a constat in clasificarea de imagini monochrome a tomogarafiilor la plamani. Acestea puteau fi incadrate in 3 clase, iar dimensiunea lor este de 50x50 pixeli. Pentru rezolvare am folosit mai multi clasificatori, dar in continuare voi prezenta modelele bazate pe SVM si CNN (cu o retea pre-antrenata de tip ResNet50).

2 SVM (Masini cu vectori suport)

Unul dintre clasificatorii pe care i-am incercat este SVC. Acesta este un clasificator binar, principiul de functionare fiind acela ca el pentru o clasa incearca sa construiasca o delimitare fata de celelalte clase care sa maximizeze numarul de date clasificate corect.

Pentru citirea datelor, folosesc modulul PIL, urmand ca dupa ce apelez functia asarray din numpy, acestea sa fie transformate in array-uri de 50x50. *Trasaturile* datelor sunt date de pixelii imaginilor. Prima modificare a fost aceea de a liniariza datele, folosind functia flatten(), transformandu-le in array-uri unidimensionale de marime 2500, cu valori intre 0 si 255.

Pentru implementare, am folosit clasificatorul SVC din modulul svm, libraria sklearn. Implementarea din sklearn are o abordare one-vs-one (se antreneaza cate un clasificator pentru fiecare pereche de clase, eticheta finala fiind cea care obtine cat mai multe voturi pe baza acestor clasificatori).

Hiperparametrii pe care i-am ajustat (folosind random search) au fost:

- Parametrul de regularizare C care indica modelului cat de mult sa sa evite clasificarea gresita; un C mare duce ca modelul sa incerce sa clasifice gresit cat mai putine date, marginile fiind foarte restranse (overfitting); un C mic duce modelul la a lasa o marja mare de eroare la margini, rezultand intr-un numar mai mare de date clasificate gresit (underfitting)

- Parametrul *kernel* care reprezinta tipul de kernel folosit in clasificarea datelor. Rolul acestuia este de a proiecta datele in spatii care pot fi liniar separabile pentru model, urmand ca relatiile liniare sa fie cautate in noul spatiu

C=1 (default)	C=2.1	C=3	C=3.5	C=3.7	C=4
0.486	0.496	0.498	**0.501**	0.500	0.4966

Pastrand C = 3.5, am obtinut urmatoarele rezultate pentru diferitele tipuri de kernel:

kernel='rbf' (default)	kernel='linear'	kernel='poly'	kernel='sigmoid'
0.501	0.372	0.454	0.362

Pe langa rezultatele obtinute, si timpul de rulare a fost afectat de kernel-ul ales, putand varia intre 10min (rbf) si 2h (linear).

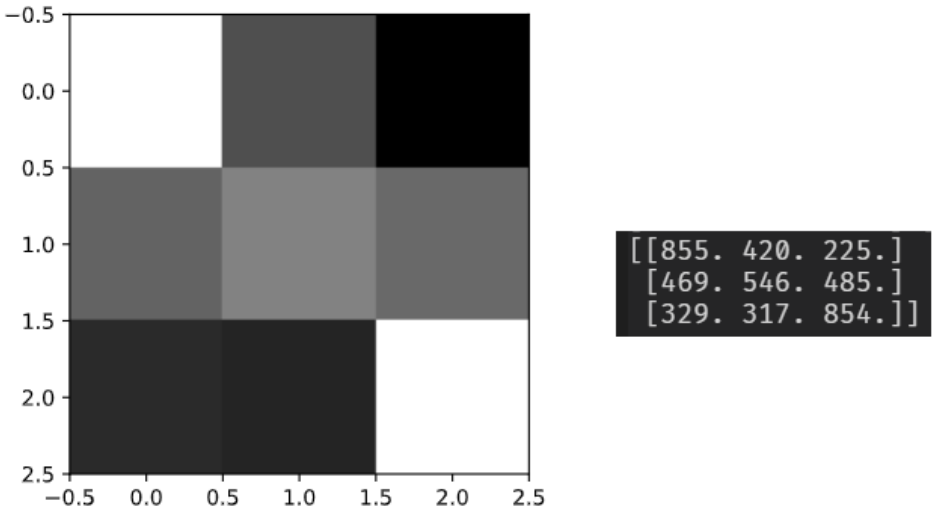


Figure 1: Matricea de confuzie

3 Convolutional Neural Network (w/ ResNet50)

O alta abordare pe care am incercat-o este aceea de a lucra cu CNN-uri. Initial am inceput cu modele la care am adaugat manual toate layer-ele, urmand ca apoi sa folosesc arhitectura ResNet50, fiind o retea preantrenata. Pentru primul tip de modele, am reusit sa obtin acuratete de 0.75 - 0.80.

In continuare voi detalia modelul bazat pe ResNet50.

ResNet50 este o retea convolutionala care are 50 de layere, dintre care 48 de layere convolutionale si 2 layere de pooling, unul de tip MaxPool si altul de tip AveragePool.

Un layer de tip Convolutional are rolul de a extrage trasaturile din input si a invata modelul cu acestea, urmand ca outputul dat de acestea sa fie redus si prelucrat de catre straturile de tip Pooling. AveragePool ia valoarea medie de pe un kernel cu o dimensiune patratica, dat de convolutionale, in timp ce MaxPool ia valoarea maxima.

Citirea datelor este asemanatoare cu cea de la modelul SVC, doar ca deoarece ResNet este configurat pe imagini de 224x244, am ales sa adaug un strat in model care sa imi faca resize la imagini pentru dimensiunile potrivite (renuntand la apelarea functie flatten odata cu citirea). De asemenea, la citire am transformat datele in imagini de tip RGB, din aceleasi considerente.

Dupa ce inputul a fost prelucrat de catre modelul preantrenat ResNet, am adaugat cateva straturi aditionale pentru a duce la o imbunatatire a acuratetii (parametrul include_top are rolul de a determina

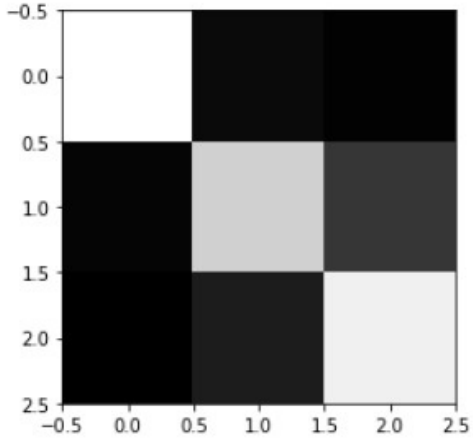
daca ultimul fully-connected layer din ResNet sa fie activat). Primul nou strat este Flatten, care are rolul de a liniariza outputul intr-un array unidimensional. Peste acestea am adaugat la final straturi de tip Dense si Dropout.

Cele de tip Dense sunt straturi de neuroni fully-connected la input si output, pentru care am ales ca functie de activare ReLu cu definitia y = max(0, x), un avantaj reprezentandu-l usurinta si rapiditatea de a fi calculata. Dupa mai multe incercari de a modifca parametrii, am putut observa faptul ca marind numarul de neuroni de pe straturile dense sau scazandu-l, nu a dus la o modifcare semnificativa a acuratetii pe datele de validare (pentru 1000 neuroni pe primul strat cu 500 pe al 2-lea, acuratetea a ramas aproximativ aceeasi, 0.858).

Straturile Dropout au rolul de renunta in mod aleator la trasaturi invatate prealabil, reducand riscul de overfitting. Ultimul strat are o functie de activare de tip softmax care prelucreaza array-ul dat ca input si il transofrma in array de probabilitati pentru fiecare clasa.

De asemenea, am ales sa folosesc un learning rate adaptiv (dupa a 10-a, respectiv a 13-a epoca, learning rate-ul scade pentru a imbunatati acuratetea, obtinand o diferenta de 0.03 in plus). Ca optimizator am ales Adam care simuleaza algoritmul coborarii pe gradient (gradient descent).

Inainte de a antrena datele, impart setul de date (antrenare si validare) la 255 pentru ca valorile datelor sa fie cuprinse in intervalul [0, 1], imbunatatind timpul de executie a clasificatorului.




```
[[1412.0, 68.0, 20.0]
[41.0, 1152.0, 307.0]
[9.0, 163.0, 1328.0]]
```

Figure 2: Matricea de confuzie

O alta configuratie pe care am incercat-o a fost aceea de a avea 4 strasturi Dense (cu tot cu ultimul) si Dropout si batch_size=64, numarul de epoci fiind tot 20. Aceasta mi-a adus o acuratete de 0.81. Dupa ce am injumatatit batch_size-ul si am scos cele 2 layere acuratetea mi-a crescut, dar si timpul de rulare.

In functie de parametrii dati la Dropout, am obtinut mici modificari in acuratete: in cazul in care marea foarte mult Dropout-ul riscam sa fac underfitting la model si sa imi scada acuratetea cu 0.02 - 0.04, iar in cazul opus, riscam sa fac overfitting la model.

4 Concluzii

Din modelele pe care le-am incercat, aproape cu toate am reusit sa obtine o predictie de peste 40%, cea mai buna acuratete pe datele de validare fiind data de CNN-uri (∼85%), acestea fiind specifice image classification.