Brain Anomaly Detection

1. Prezentarea task-ului:

In cadrul proiectului a trebuit sa clasificam un set de imagini in 2 clase, care reprezinta prezenta(aparinand clasei 0), respectiv absenta(apartinand clasei 1) anomaliilor pe creier. Datele primite, imagini de dimesiune 224 x 224, au fost imparite in 3 categorii. Date de train (15 000 de imagini),care au asignat un label. Pe acestea le-am folosit pentru a antrena modelul. Datele de antrenare au rolul de a invata modelul de inteligenta artificiala pattern-uri care mai apoi ajuta la a face preziceri pe date noi. A doua categorie o prezinta datele de validare (2000 de imagini), care si acestea au un label asignat. Pe acestea le-am folosit pentru a monitoriza acuratetea prezicerilor dupa invatare. Fiind date noi, pe care modelul inca nu le-a vazut, am putut calcula f1_score, o unitate de masura ce calculeaza performanta modelului. In functie de f1_scor imi dadeam seama daca modelul facea over-fitting sau nu, si in acelasi timp observam daca modificarile aduse modelului rezulta in niste preziceri imbunatatite. A treia categorie o reprezinta datele de testare (5149 de imagini). Pe aceste imagini a trebuit sa facem predictiile finale, cele pe care le-am submis pe kaggle.

2. Modelele folosite:

Pentru acest task am folosit 2 modele: Naive Bayes si CNN(Convolutional Neural Network)

- ## Naive Bayes:

Naive Bayes este un algoritm de clasificare care se bazeaza pe teorema lui Bayes de probabilitati conditionate.

P(label | caracteristici imagine) = P(caracteristici imagine | label) * P(label) / P(caracteristici imagine) Formula folosita in clasificarea imaginilor.

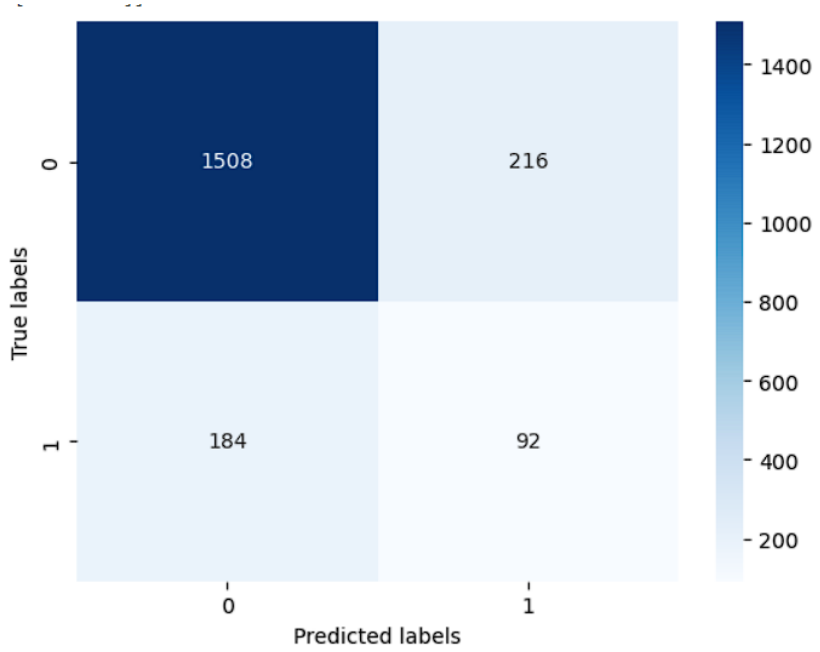
Chiar daca Naive Bayes nu este optim pentru a clasifica imagini, acesta poate fi folosit. Pentru clasificarea imaginilor algoritmul interpreteaza fiecare pixel. Deoarece setul nostru de date este reprezentat de imagini gray-scale, pixelii imaginilor reprezinta gradul de luminozitate. In timpul antrenarii, modelul calculeaza probabilitatile carcateristicilor pentru clasele in care acestea trebuiesc impartite. Aceste probabilititati sunt de tipul: pentru un ct-scan ce contine anomalii probabilitatea este mai mare de a avea pixelii cu un anumit grad de luminozitate intr-o zona. Atunci cand efectueaza predictii, se foloseste de formula lui bayes. In functie de rezultat, se alege clasa cu probabilitatea cea mai mare din care imaginea face parte.

Naive Bayes nu este eficient pentru clasificarea imaginilor deoarece algoritmul considera ca carecteristicile imaginilor sunt independente fata de alte caracteristici ale imaginii. Acest lucru nu este adevarat deoarece intr-o imagine formata din mai multi pixeli, toti pixelii impreuna formeaza un intreg in care depind ului de altul. In acelasi timp, Naive Bayes nu este capabil sa captureze relatii complexe intre imagini.

In imlpementarea mea m-am folosit de libraria OpenCv pentru a incarca imaginile. Initial le salvez intr-o lista, pe care mai apoi o transform intr-un numpy array. Fiind primul model incercat nu am abordat tehnici de augmentare a datelor. Deoarece le incarc pe toate o data si nu folosesc alte metode precum generatoare, nu am putut sa normalizez datele (prin

impartirea la 255), deoarece ieseam din memoria RAM. Am avut aceeasi abordare pentru datele de validare. Numpy array-urile mai apoi le voi schimba “forma” (reshape) pentru a le putea da drept parametru modelului. Ma folosesc de modelul GaussianNB din libraria skit-learn. Acest model se bazeaza pe faptul ca pixelii sunt distribuiti in concordanta cu Gaussian Distribution. Dupa aceea calculez f1_scorul obtinut. Cel calculat de mine pe datele de validare este de 0.315, iar punctajul obtinut pe platforma este 0.359 (dupa finalizarea competitiei). Deoarece acest algoritm nu este optim, urmatorul model ales este CNN

Matricea de confuzie:



- **CNN:**

Convolutional Neural Network este un algoritm de deep learning care este in special folosit pentru procesare de imagine si de video. Algoritmul prelucreaza inputul(setul de imagini in acest caz) cu ajutorul straturilor care-l alcatuiesc.

Modul de functionare al algoritmului:

Statul de input este cel care preia setul de imagini. Stratul Convolutional aplica filte, numite Kernell, care au rolul de a identifica caracteristici distincte ale imaginilor cum ar fi: margini, colturi, luminozitate etc. Functia de activare este aplicata fiecarui output produs de neuroni. In cadrul proietului m-am folosit de cea mai comun folosita functie, si anume ReLU (Rectified Linear Unit). Statul de pooling primeste outputul functiei de activare. Acesta extrage cele mai importante caracteristici din setul de date. Acesti pasi se repeta pentru a lasa modelul sa invete in adancime pentru a reusi clasificarea pe setul de date. Este o practica buna ca fiecare strat nou convolutional sa primeasca mai multe filtre decat cel anterior pentru a creste complexitatea invatarii. Stratul de Flatten primeste output-ul ultimului strat convolutional si il transforma intr-un vector, pe care mai apoi il va transmite straturilor Fully Connected, cele care

in final vor efectua clasificarea. Obiectivul principal este de a minimiza functia de loss, cea care masoara diferentele dintre output-ul prezis si cel corect.

Pentru model meu de CNN, m-am folosit de Keras, high-level neural network API. L-am ales pe acesta deoarece are un design user-friendly si relativ simplu de folosit. In Keras, un model este compus de layere, suprapuse sub forma de stiva, ceea ce inseamna ca output-ul unui strat va fi imputul pentru urmatorul.

Model meu:

Consta in 4 straturi convolutionale, printre care adaug straturi de MaxPooling, Batch Normalization si de Dropout, urmat de un start de Global Average Pooling si un Fully Conected layer cu functia de activare sigmoid.

Primul strat este cel de input. Aici specific dimensiunea imaginilor (224 x 224) si respectiv tipul acestora, gayscale input_shape=(224, 224, 1)

Primul strat convolutional are 64 de filtre de marimea 3 x 3. Kernell-ul este o matrice care scaneaza input-ul, iar apoi efectueaza produsul scalar intre weight si valoarea pixelilor. Aceste tipuri de straturi produc feature maps. Folosesc ReLU drept functia de activare. Padding-ul setat la same mentine aceasi forma a output-ului ca si cea a input-ului.

Stratul de Batch Normalization normalizeaza output-ul stratului anterior mentionat prin scaderea mean-ului si impartirea la devierea standard a batch-ului.

Stratul de Max Pooling este un strat care reduce dimensiunile outpu-ului din stratul anterior, luand valoare maxima intre o fereasta de (2 x 2).

Stratul de Drop Out este o metoda de regularizare, care in mod aleator seteaza in timpul antrenamentului o fractiune din unitatile de input la zero, cu o rata de 0,25.

Dupa repet acest proces, schimband insa numarul de filtre. Urmatoarele vor avea, in aceasta ordine 128, 256 si 512 filtre.

Ultimul strat convolutional, cel cu 512 filtre foloseste Global Average Pooling 2D. Acesta calculeaza media pentru fiecare canal din feature map, valoarea pentru acesta fiind transmisa ultimului strat, cel de clasificare.

Layer-ul dense este fully conected cu un singur output, ce foloseste functia de activare sigmoid, care este folosita pentru clasificari binare.

Procesarea Inputului:

Pentru incarcarea imaginilor m-am folosit de Panda Dataframe, care este o structura de date bidimensioanala ce are de asemenea si label-uri. Aceasta seama cu un tabel in care datele sunt organizate in randuri si coloane. Aici am stocat path-urile imaginilor cat si label-urile asociate fiecareia.

Apoi am folosit Image Data Generator pentru a putea efectua augmentari de date pentru imaginile de train. Prin rescale=1./255 am normalizat pixelii imaginilor, impartindu-i pe fiecare la valoarea 255, avand valoarea noua cuprinsa intre 0 si 1. Aceste imagini mai apoi au fost transformate in grayscale. Am dezactivat functia de shuffle pentru ca atunci cand o foloseam f1_scorul obtinut nu trecea de base line, obtinand 0,11. Acest lucru este probabil din cauza imbalantei dintre clase. Imaginile care nu prezentau anomalii erau intr-un numar mai mare

decat celelalte. Amestecarea lor poate duce la o distribuire a claselor. Imaginile le-am trimis catre model, in batch-uri de 32.

Class Imbalance:

Asa cum am mentionat mai sus, cele 2 clase de nu sunt impartite egal. Modelul va prezice mai de graba label-uri din clasa majoritara. Spre exemplu daca 90% dintre imagini nu au anomalii, modelul le poate clasifica pe toate cu label-ul 0 avand o acuratete de 90%, dar de fapt nu a invatat nimic despre input-ul primit.

Pentru a rezolva aceasta problema m-am folosit de class weighting. Am asignat un weight mai mare clasei minoritare, folosindu-ma de compute_class_weight() din scikit-learn.

Compilarea Modelului:

Am folosit binary cross-entropy pentru functia de loss, optimizatorul Adam, antrenand in 20 de epoci. Prima data am antrenat in 15 epoci, insa am obtinut un scor mai mic, obtinand f1_score de 0,596. Pentru acest model final, am obtinut scorul de 0.6708.

Alte modele/tehnici abordate:

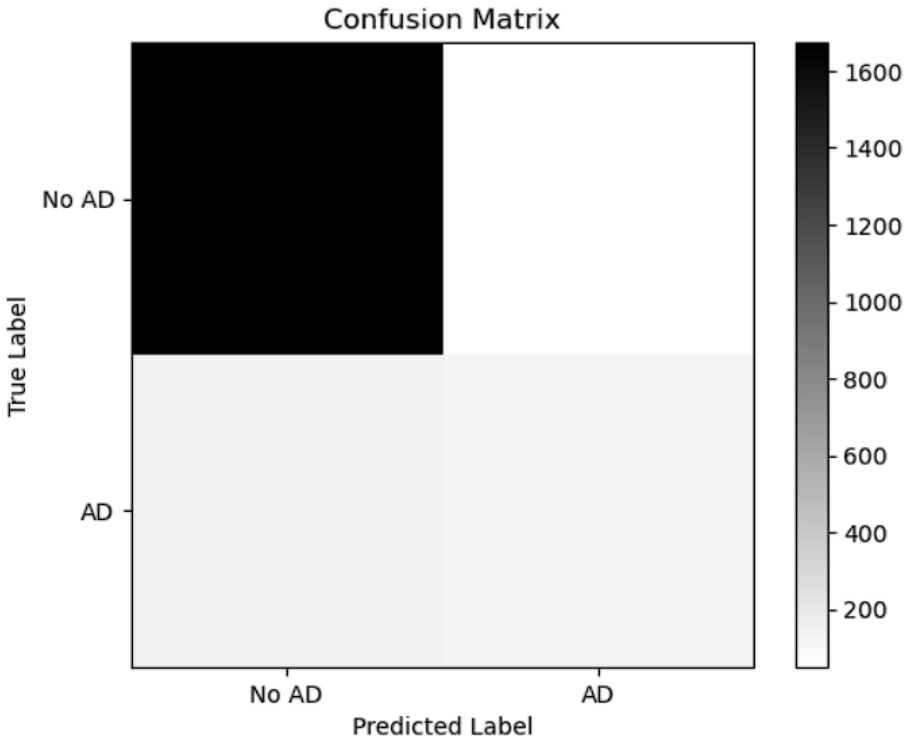
In primul model incercat am definit numai layerele convolutionale si cele de Average Pooling si Dense, folosid tot generaoare pentru imagini, insa f1_scorul a fost sub base-line. Dupa ce am ajuns la varianta finala mi-am dat seama ca default-ul pentru shuffle din Image Data Generator este de de True, ceea ce de asemena juca un rol in scorul mic obtinut.

Am renuntat apoi la Generatoare, nestiind ca acel shuffle influenteaza acuratetea. Le-am incarcat la fel cum am facut si la Bayes, folosit OpenCV. Nu am efectuat augmentari de date, dar am adaugat si celelalte straturi de normalizari si dense, dar si parametru de Strides cu valoare de (1,1) pentru primul layer Convolutional, respectiv (2,2) pentru celelalte. Acest lucru a crescut f1_scorul la 0,49.

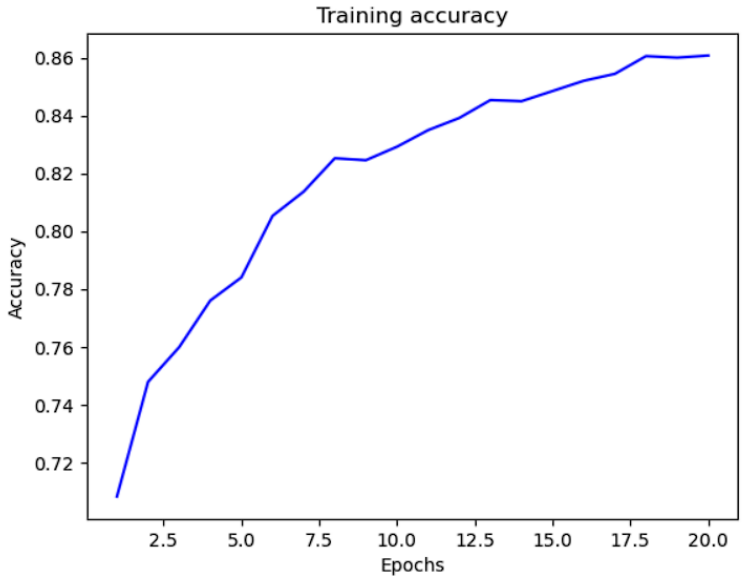
Am revenit la Image Generator pentru a putea folosi augmentari de date. Am incercat sa adaug featurewise_center=True, featurewise_std_normalization=True, horizontal_flip=True, vertical_flip=True, iar scorul a crescut la 0,52.

Dupa mai multe incercari am observat ca cea mai buna varianta renunta la parametrul de strides din layere si singurele augmentari de date pe care le face sa fie normalizarea, adaugand insa si weight pe clase. Scorul final pe care l-am obtinut este de 0,67 (dupa inchiderea competitiei)

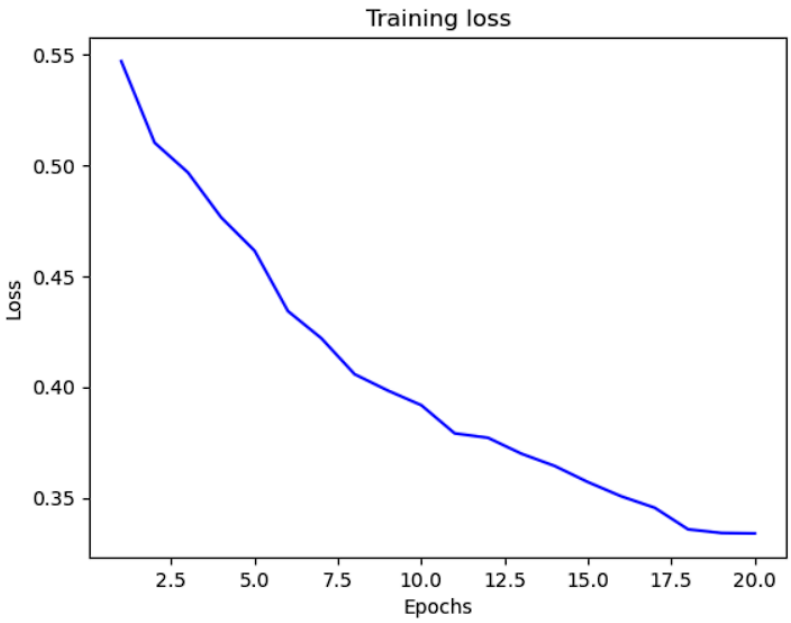
Matrice de confuzie:



Acuratetea in timpul antrenamentului:



Loss-ul in timpul antrenamentului:


Training loss
0.55
0.50
0.45
0.40
0.35
Loss
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Epochs