Muscalu Diana
Grupa 244

Brain Anomaly Detection

I. Descrierea Proiectului

Tema proiectului a fost clasificarea unor imagini cu radiografii la creier, de dimensiune 224 x 224 pixeli, în două clase distincte și anume 0 (fără anomalie) și 1 (cu anomalie).

II. Setul de date

- 15.000 de imagini de antrenare cu label-urile asociate
- 2.000 de imagini de validare cu label-urile asociate
- 5.149 de imagini de test

III. Abordări în rezolvarea proiectului

- NB
- RF
- CNN

A. Gaussian Naive Bayes (GaussianNB)

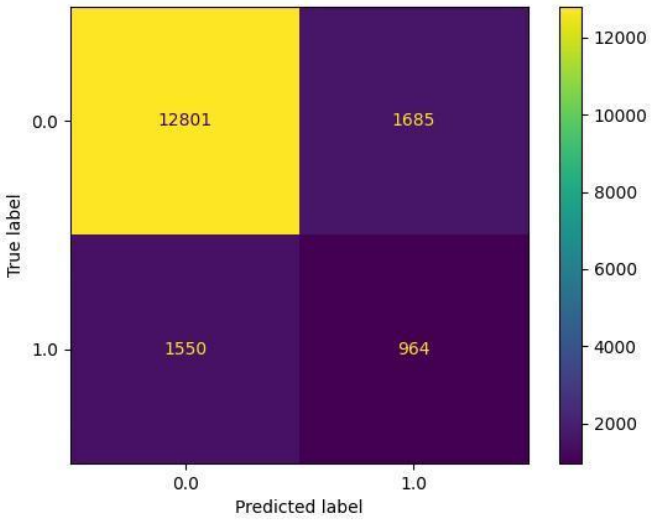
O primă abordare în rezolvarea problemei a fost Gaussian Naive Bayes. Acesta este un model simplu ce clasifică diferite obiecte pe baza unor caracteristici, pe care el le consideră independente, de aceea fiind "naiv". Se folosește de o distribuție gaussiană a probabilităților pentru a estima probabilitatea ca un obiect să aparțină de o categorie. Acestea se determină pe baza probabilităților marginale ale fiecare caracteristici și a probabilităților condiționate date de clasa de care aparține obiectul.

Pentru a citi datele m-am folosit de modulul PIL. Imaginile sunt transformate în array-uri de 64 x 64, urmând să fie transformate în array-uri unidimensionale cu ajutorul funcției flatten().

Pentru prima încercare a acestui model am decis să o păstrez cât mai simplă pentru a observa cum se comportă. Ulterior pentru a avea un set mai mare de antrenare și pentru a avea o performanță mai bună, am decis să îmi combin datele de antrenare cu cele de validare.

Tipul datelor	Precision	Recall	F1-Score
Simple	0.32	0.36	0.34
Combinate	0.36	0.38	0.37

Din acest tabel, determinat de funcția classification_report(labels, predictions), se poate observa ca unind imaginile de antrenare cu cele de validare, modelul nostru ajunge la o performanță mai bună pe datele de test. F1-Score este cel mai important de urmărit pentru a vedea performanța deoarece ia în calcul ambele aspecte: Precision și Recall.



Matricea de confuzie pentru Gaussian NB

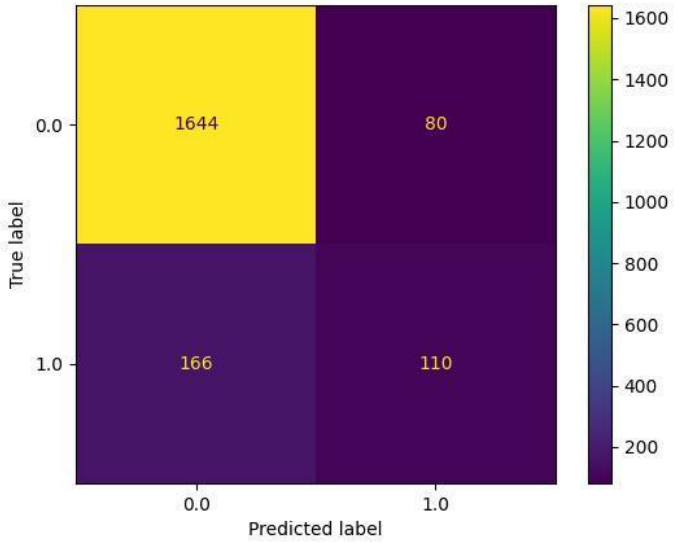
B. Random Forests (RF)

Următoarea abordare aleasă pentru rezolvarea problemei a fost Random Forests. Aceasta este o metodă de clasificare care se bazează pe un ansamblu de arbori de decizie. Fiecare arbore este construit utilizând seturi diferite de date, alese aleatoriu din setul de antrenare și totodată fiecare dintre aceștia oferă o predicție individuală. Rezultatele sunt după aceea combinate prin votare pentru a se ajunge la o predicție finală cât mai bună.

Aici am decis să îmi păstreze datele combinate pentru a avea un set mai mare de date de antrenare. Inițial ca și la Gaussian Naive Bayes am ales să păstrez o abordare simplă pentru a vedea cum se comportă modelul. După aceea am folosit un GridSearchCV pentru a căuta hiperparametrii optimi pentru model.

GridSearchCV	Precision	Recall	F1-Score
NU	0.58	0.40	0.47
DA	0.60	0.41	0.48

Din tabelul de mai sus reiese faptul că modelul nostru este mai bun pe varianta cu GrisSearchCV, însă diferența dintre cele două este foarte mică. Această diferență mică s-a păstrat și în submisiile de pe Kaggle unde la final am ajuns la acuratețea de 0.51428 pentru RF fără parametrii optimii, respectiv 0.51921 pentru RF cu parametrii optimizați.



Matricea de confuzie pentru RF

C. Convolutional Neural Network (CNN)

CNN este o rețea neuronală convoluțională care cuprinde straturi de convoluție cu diferite filtre și peste care se aplică niște funcții de activare.

Aceasta este ultima abordare pe care am încercat-o în cadrul proiectului.

Inițial am avut am încercat să plec cu un model de bază, pe care să îl modific pe parcurs pentru a vedea ce merge și ce nu merge pe acesta. Am decis ca pentru prima variantă să nu fac augmentare pe imagini, lăsându-le în forma lor, reducându-le doar la dimensiunea de 64 x 64 pixeli și împărțind la 255 pentru a realiza o normalizare. Pentru implementarea rețelei m-am folosit de keras din tensorflow. Modelul folosit este unul secvențial care are rolul de a lua datele de ieșire de la stratul anterior și a le folosi ca date de intrare pentru următorul.

Ulterior am apelat la augmentarea de imagini pentru a obține o acuratețe mai bună și am folosit ImageDataGenerator.

În continuare voi prezenta mai detaliat layerele și hiperparametriipe care i-am folosit:

- Conv2D - strat de convoluție ce extrage caracteristi din imagini ce le transpune într-o matrice
    i. Filters – am încercat mai multe variante și am folosit un număr de filtre variabil și anume: 16, 32, 64, 128
    ii. Kernel_Size – am folosit ca parametru un kernel de (3, 3)
    iii. Input_Shape – am folosit (64, 64, 3) adică o imagine de dimensiune 64 x 64 RGB
    iv. Activation – am folosit “relu” deoarece este un activator eficient
- MaxPooling2D – strat ce reduce dimensiunea hărții de caracteristici determinate în urma convoluției și ajută la extragerea caracteristicilor mai importante

- BatchNormalization – strat folosit pentru normalizarea valorilor de activare dintr-un strat anterior și are ca scop accelerarea procesului de antrenare
- Flatten – strat ce aplatizează input-ul pentru straturile Dense care au nevoie ca datele să fie într-un format unidimensional
- Dropout – strat ce ajută la regularizarea rețelei neuronale și reducerea overfitting-ului prin a lăsa inactivi unii neuroni în timpul antrenării
    i. Rata a variat de la 0.3 pana la 0.55, însă am realizat că cea mai bună performanță a fost cu rata de 0.45
- Dense - numit și Fully Connected Layer, e un strat ce primește intrări de la stratul anterior și calculează ieșirile finale
    i. Aici am încercat dacă este mai bine să am doar un strat Dense final de 1 sau dacă să mai am unul înainte, iar în final am ales varianta cu două straturi Dense, iar primul are ca parametru 64
    ii. Ultimul strat Dense are ca parametrii 1 și activare “sigmoid” deoarce avem o clasificare binară, iar funcția sigmoid comprimă valoril de ieșire în intervalul [0, 1]

Mai departe voi prezenta ce am folosit pentru compilarea modelului:

- Funcția de loss pe care am folosit-o este BinaryCrossentropy care este folosită pentru clasificarea binară și compară valorile prezise de model cu cele reale și determină o măsură a diferenței dintre acestea
- Optimizatorul folosit este “adam” cu un learning rate de 0.01

Acum voi prezenta modelele folosite și acuratețea obținută cu acestea:

Strat	Hiperparametrii
Conv2D	filters=32, kernel=(3,3), activation=relu
BatchNormalization	default
MaxPooling2D	default
Conv2D	filters=64, kernel=(3,3), activation=relu
BatchNormalization	default
MaxPooling2D	default
Conv2D	filters=128, kernel=(3,3), activation=relu
BatchNormalization	default
MaxPooling2D	default
Flatten	default
Dropout	0.45
Dense	1, activation=sigmoid

Cu acest model de CNN am obținut o acuratețe de 0.53. De aici am modificat rețeaua și am ajuns la următoarea sa formă și anume:

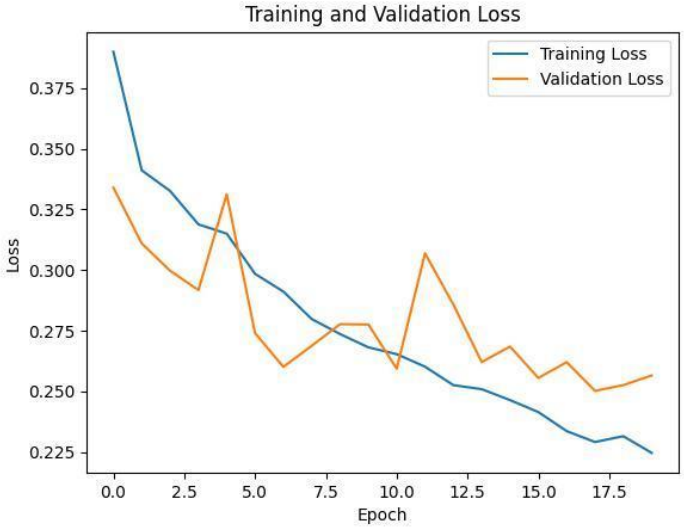
Strat	Hiperparametrii
Conv2D	filters=32, kernel=(3,3), activation=relu
MaxPooling2D	(2,2)
Conv2D	filters=64, kernel=(3,3), activation=relu
MaxPooling2D	(2,2)
Conv2D	filters=128, kernel=(3,3), activation=relu
MaxPooling2D	(2,2)
Conv2D	filters=128, kernel=(3,3), activation=relu
MaxPooling2D	(2,2)
Flatten	default
Dense	64, activation=relu
Dropout	0.45
Dense	1, activation=sigmoid

Prin mai multe încercări am ajuns la această variantă observând din aproape în aproape că în contextul problemei straturile de BatchNormalization nu ajutau la creșterea acurateței deoarece aveam batchsize-urile destul de mici. În plus am mai adăugat un strat de Conv2D, MaxPooling2D și încă unul Dense, observând că funcționează mai bine pentru această clasificare. Acest model de CNN a dat o acuratețe de 0.60.

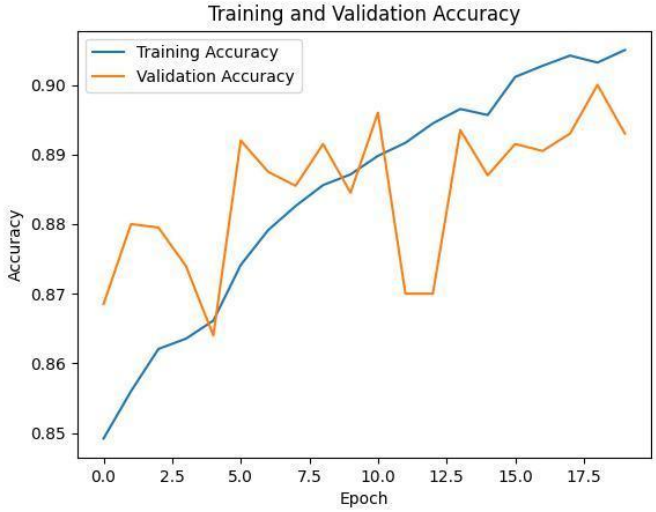
La final pentru a obține o acuratețe mai mare am apelat la augmentarea datelor, realizată cu ajutorul ImageDataGenerator. Astfel am folosit următorii parametrii:

- shear_range = 0.2 – interval în care imaginea poate fi înclinată pe axa orizontală
- zoom_range = 0.1 – interval în care imaginea poate fi mărită sau micșorată
- horizontal_flip = True – 50% șanse ca imaginea să fie răsucită orizontal

Cu aceste modificări și cu modelul antrenat pe 20 de epoci, am ajuns la rezultatul final din cadrul competiției de 0.649.



Graficul pentru Loss pe Train și Validation



Graficul pentru Accuracy pe Train și Validation

IV. Concluzii

După încercarea mai multor modele pentru clasificarea imaginilor, cel mai bun a fost CNN la care am adăugat o augmentare pe datele de train, cu care am reușit să obțin cea mai bună acuratețe din cadrul competiției.

V. Bibliografie

- Laboratoare
- https://scikit-learn.org/stable/supervised_learning.html#supervised-learning

- https://www.tensorflow.org/api_docs/python/tf/keras/
- https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
- https://keras.io/api/