Brain anomaly detection

Inteligență artificială

Gheorghe Robert-Mihai

2022-2023

1 Despre proiect

Tema proiectului este clasificarea unor imagini de scanări CT ale creierului în două clase, una care conține anomalii și una care este normală. Imaginile sunt la rezoluția de 224x224 pixeli pe canalul grayscale.

Setul de date conține 22149 de imagini împărțite astfel:

- Date de antrenare → 15000 de imagini cu label-urile asociate
- Date de validare → 2000 de imagini cu label-urile asociate
- Date de testare → 5149 de imagini

2 Modele ML folosite

I. Naïve Bayes (NB)

II. Convolutional Neural Network (CNN)

I. Naïve Bayes

Prima abordare a fost un model simplu de GaussianNB. Acesta se bazează pe Teorema Bayes și presupune că datele de antrenare sunt într-o distribuție Gaussiană. Teorema Bayes spune că probabilitatea unei ipoteze condiționată la un set de date poate fi calculată prin înmulțirea probabilității anterioare a ipotezei cu probabilitatea datelor condiționată la ipoteză, rezultat împărțit mai apoi la probabilitatea datelor. Matematic, formula se reprezintă astfel:

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

unde

$$\begin{aligned} P(A|B) &= \text{probabilitatea condiționată a ipotezei } A \text{ dat setul de date } B \\ P(B|A) &= \text{probabilitatea condiționată a datelor } B \text{ dată ipotezei } A \\ P(A) &= \text{probabilitatea anterioară a ipotezei } A \\ P(B) &= \text{probabilitatea datelor } B \end{aligned}$$

Fiind un model naiv m-am folosit doar de librăria PIL pentru a încărca imaginile și pentru a converti array-urile în unele de pixeli grayscale, apoi am folosit modelul GaussianNB din librăria scikit-learn. Această abordare a avut un scor de ≈ 0.348. Matricea de confuzie este următoarea:

$$\begin{bmatrix} 0.89125296 & 0.7012987 \\ 0.10874704 & 0.2987013 \end{bmatrix}$$

II. Convolutional Neural Network

O rețea neuronală convoluțională este un model împărțit pe mai multe straturi de convoluție ce conțin neuroni care reprezintă funcții aplicate pe datele noastre pe care i le pasăm modelului. Mi-au fost utile librăriile tensorflow și keras pentru acest tip de model și pentru straturile acestuia.

1. Preprocesarea datelor

Pentru această abordare am considerat că datele au nevoie de o preprocesare suplimentară pentru o acuratețe sporită, dar și pentru a folosi mai puțină memorie. Operațiunile folosite pe imagini în acest pas:

- transformarea în grayscale
- resize pe dimensiunea (50,50) (pentru a se încadra în RAM)
- aplicarea unui Gaussian Blur
- aplicarea de threshold pe imaginea blurată
- erodarea și dilatarea imaginii
- în final am putut găsi contururile relevante din imagine care puteau semnala o tumoare

2. Augmentarea datelor

Pentru a mări acuratețea a fost nevoie de augmentare de date pentru a extinde și diversifica setul de date. Din fiecare imagine din setul de antrenare (și validare) și testare s-au mai născut încă 2 imagini, astfel:

- o imagine pusă în oglindă (flip orizontal)
- o imagine înclinată la 45° spre dreapta

Pe aceste imagini s-a aplicat de asemenea preprocesarea de la pasul precedent.

3. Straturi folosite

- Conv2D $\rightarrow$ strat care genereza o matrice de features din input folosind o funcție kernel de convoluție. Funcția de activare folosită este ReLU
- MaxPooling2D $\rightarrow$ strat ce aplică algoritmul ferestrei glisante peste input pe o dimensiune specificată ca parametru, reducând dimensiunile cât de mult posibil pentru a putea găsi legături între features
- Flatten $\rightarrow$ strat de aplatizare a inputului pentru a-l transforma într-un vector unidimensional
- Dense $\rightarrow$ strat dens conectat în rețeaua neuronală. Funcția de activare folosită este *sigmoid* deoarece avem parte de clasificare binară

- BatchNormalization → strat ce normalizează inputul în fiecare batch pentru a standardiza învățarea
- Dropout → strat care dezactiveaza random (în funcție de parametrul ales de noi) câțiva neuroni într-o epocă. Acest layer este util pentru a reduce overfitting-ul

4. Compilarea modelului

- **Funcția de pierdere:** *binary_crossentropy* deoarece modelul clasifică binar clasele
- **Funcția de optimizare:** *adam* deoarece este un algoritm mai puternic de Stochastic Gradient Descent
- **Metrică:** acuratețea

5. Callbacks

Am folosit Callbacks pentru analiza anumite date pe parcursul finalizării fiecărei epoci în parte. Am considerat că sunt necesare următoarele:

- **Early Stopping** pentru a putea opri învățarea mai devreme în cazul în care modelul nu mai învață nimic timp de un număr ales ca parametru de epoci
- **Tensorboard** pentru a puea vizualiza la final evoluția modelului pe parcursul epocilor în Tensorboard, care este o librărie legată de Tensorflow pentru vizionarea mai detaliată a procesului de învățare

6. Ajustarea hiperparametrilor

Hiperparametrii pe care a trebuit să îi testez și ajustez prin testarea repetată a modelului am considerat că sunt:

- **Learning rate**-ul funcției Adam pentru a evita underfitting-ul și overfitting-ul
- **Numărul de epoci** pe care îl tot creșteam până intervenea callback-ul de *Early Stopping*
- **Batch size-ul** pentru a vedea care este numărul optim de imagini de trecut prin rețea simultan pentru un rezultat mai satisfăcător

Mai jos se pot observa scorurile opținute pe următoarele seturi de hiperparametrii:

Learning rate	**Număr de epoci**	**Batch size**	***Scor*** (local)
0.001	40	50	0.51
0.01	40	50	0.49
0.001	45	50	0.44
0.001	40	60	0.52
0.001	40	65	0.53
0.001	45	65	0.56
0.001	50	65	0.47
0.00001	55	70	0.53

Ultimul set de hiperparametrii a fost cel optim.

7. Variantă finală

În final, modelul are următoarea structură:

Layer (type)	**Output shape**	**Param**
conv2d (Conv2D)	(None, 50, 50, 32)	320
max_pooling2d (MaxPooling2D)	(None, 25, 25, 32)	0
conv2d_1 (Conv2D)	(None, 25, 25, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 12, 12, 64)	0
conv2d_2 (Conv2D)	(None, 12, 12, 128)	73856
conv2d_3 (Conv2D)	(None, 12, 12, 128)	147584
max_pooling2d_2 (MaxPooling2D)	(None, 6, 6, 128)	0
flatten (Flatten)	(None, 4608)	0
dense (Dense)	(None, 256)	1179904
batch_normalization (BatchNormalization)	(None, 256)	1024
dense_1 (Dense)	(None, 128)	32896
dropout (Dropout)	(None, 128)	0
batch_normalization_1 (BatchNormalization)	(None, 128)	512
dense_2 (Dense)	(None, 1)	129

Matricea de confuzie a acestui model arată astfel:

$$\begin{bmatrix} 0.93202147 & 0.49845201 \\ 0.06797853 & 0.50154799 \end{bmatrix}$$

Acest model a avut o acuratețe de ≈ 0.58503 pe Kaggle.

Câteva detalii din Tensorboard:

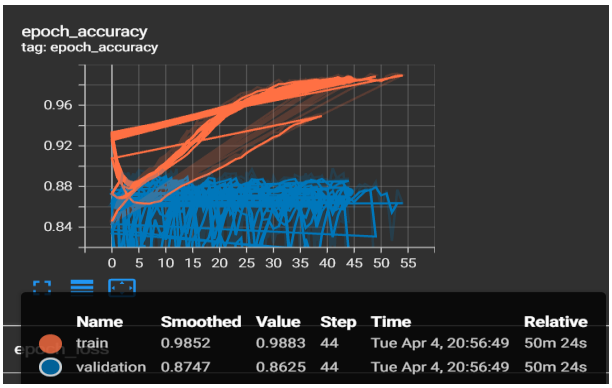

Figure 1: Epoch accuracy

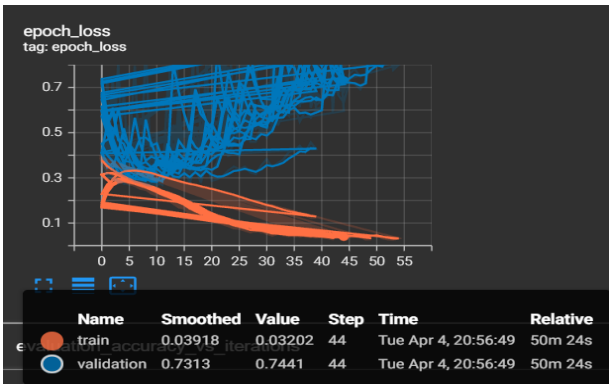

Figure 2: Epoch loss

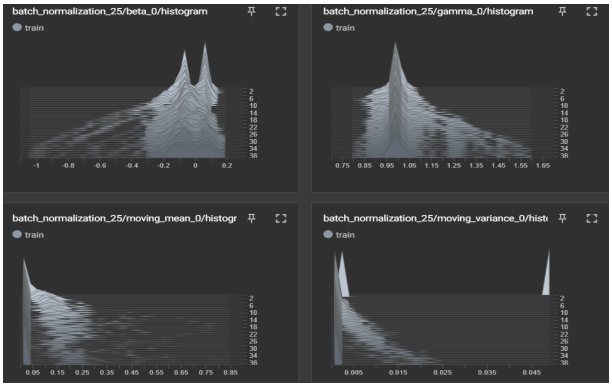

Figure 3: Batch Normalization