Anul 2 - Semestrul 2
Inteligență Artificială (ML)
Documentație proiect

Beșliu Radu-Ștefan
Grupa 243

Cuprins

1 Descrierea proiectului

Acest proiect constă în clasificarea unori imagini de tip tomografic. Formatul este de 224x224 pixeli, grayscale. Există două clase în care pot fi încadrate: 0, pentru cele care nu prezintă anomalii; 1, pentru cele care prezintă anomalii.

Datele folosite în dezvoltarea proiectului sunt alcătuite din 15000 imagini de antrenare, 2000 de validare si 5149 de testare.

Voi prezenta două modele folosite pentru clasificarea acestora: K-nearest neighbours (KNN) și Convolutional Neural Network (CNN).

În realizarea modelului, am folosit două metode de citire și de salvare a imaginilor. Prima metodă folosește librăria matplotlib pentru citirea acestora și numpy pentru salvarea în fișiere de tip *.npy. A doua variantă se folosește de pillow pentru citirea imaginilor, transformarea lor în grayscale și resize (atunci când este cazul).

2 KNN

- **Descriere**

  În acest proiect, algoritmul KNN este folosit pentru clasificarea imaginilor. La fiecare pas, cei mai apropiați “k” vecini sunt considerați pentru următoarea predicție.

  Am folosit KNeighborsClassifier din librăria sklearn.

- **Procesarea datelor**

  Pentru acest algoritm, au fost făcute o serie de procesări. Imaginile (training, validare si testare) au fost transformate în array-uri de o dimensiune (flattening), pentru a reduce complexitatea computațională și pentru a trata fiecare pixel ca un feature individual.

- **Alegerea parametrilor**

  Am testat 3 valori pentru parametrul “n_neighbors”, acestea fiind 5, 3, 1. Pentru toate aceste valori, parametrul “metric” a primit “euclidean”. Acesta reprezintă modul în care este calculată distanța pentru a afla cei mai apropiați k vecini (trasarea unei linii drepte).

- **Informații după validare**

  Mai jos sunt prezentate câteva tabele care conțin informații legate de precision, recall, f1-score (pentru ambele clase), accuracy (macro average, weighted average) și confusion matrix, pentru fiecare din parametrii menționați mai sus. Macro average calculează average-ul valorii de precision, recall si f1-score, fiecare clasă având același weight. Weighted average calculează aceleași lucruri, luând în considerare numărul de apariții al fiecare clase (în cazul acestui dataset, 0 apare de aproximativ 6 ori mai mult decât 1, implicit rezultând într-o antrenare mai bună a modelului pentru imaginile care nu prezintă anomalii).

metric=”euclidean”
n_neighbors=5

class	precision	recall	f1-score
0	0.87	0.98	0.92
1	0.45	0.09	0.15

Accuracy (0.86)	precision	recall	f1-score
macro avg	0.66	0.54	0.54
weighted avg	0.81	0.86	0.82

	P0	P1
T0	1694	30
T1	251	25

metric=”euclidean”
n_neighbors=3

class	precision	recall	f1-score
0	0.87	0.96	0.92
1	0.34	0.12	0.18

Accuracy (0.85)	precision	recall	f1-score
macro avg	0.61	0.54	0.55
weighted avg	0.8	0.85	0.81

	P0	P1
T0	1658	66
T1	242	34

metric=”euclidean”
n_neighbors=1

class	precision	recall	f1-score
0	0.88	0.92	0.9
1	0.33	0.24	0.27

Accuracy (0.83)	precision	recall	f1-score
macro avg	0.61	0.58	0.59
weighted avg	0.81	0.83	0.82

	P0	P1
T0	1591	133
T1	211	65

- **Concluzii**

  În urma analizei celor 3 variante ale modelului KNN, am ajuns la următoarea concluzie: un număr mai mic de vecini, dați parametrului n_neighbors, îmbunătățește performanța modelului. Ultima variantă prezentată, cea cu 1 vecin, a obținut un scor de 0.26446 pe platforma Kaggle.

3 CNN

- **Descriere**

  Convolutional Neural Network (CNN) reprezintă un model utilizat pentru task-uri de clasificare și recunoaștere de imagini. Acest model este format din mai multe “straturi” (layers), care prezintă un număr variabil de filtre.

  Layer-ele folosite în acest proiect sunt:

  - Convolutional (Conv2D): acest layer este folosit pentru extragerea de feature-uri folosind un set de filtre;
  - Pooling (MaxPooling2D): utilizat pentru a reduce dimensiunea spațială a feature map-ului, pentru a extrage cele mai importante feature-uri, reducând;
  - Fully connected (Dense): conectează toți neuronii din stratul precedent cu cei din output, aplicând weight-uri pe fiecare legătură;
  - Output: produce predicția finală bazat pe feature-urile extrase din imagini.

- **Prezentarea modelelor folosite**

  Mai jos vor fi prezentate 6 variante de CNN, împreună cu f1_score-ul pe cele 2 clase (0 si 1), matricea de confuzie, precision, recall și accuracy (macro si weighted average). În mare, modelele sunt destul de asemănătoare, diferențele fiind alcătuite din modificarea parametrilor, augmentare imaginilor (rescale total, rotatie, shift, zoom, flip random etc.), adăugarea / modificarea valorilor de dropout, normalizare, max pooling, schimbarea numărului de filtre sau adăugarea unor layere în plus.

  Fine-tuning-ul parametrilor s-a realizat prin rularea modelelor și analiza informațiilor furnizate, cum ar fi: accuracy, val_accuracy, loss, val_loss. Prin intermediul acestora, se putea observa dacă modelul converge spre o soluție optimă, dacă nu se mai poate îmbunătăți, dacă suferă de overfitting / underfitting sau dacă augmentarea datelor a condus la o complexitate prea mare pentru rețeaua furnizată (ex: val_loss extraordinar de mare după un număr relativ de mare de epoci).

  Metoda de activare folosită este ReLu (nu am testat alte variante). Pentru calcularea loss-ului în funcția de compilare a modelului am folosit “binary_crossentropy”, extrem de utilă pentru dataset-urile care au 2 clase.

- **Observații**

  Pe parcursul dezvoltării proiectului, am observat câteva lucruri:

  - Fine-tuning-ul unor parametrii, chiar dacă erau diferențe foarte mici, poate conduce la modificări drastice privind valorile menționate anterior, în special scorul de pe Kaggle;
  - Mai multe filtre / layere nu conduc neapărat la o creștere a scorului;
  - Cea mai bună variantă pentru creșterea scorului, a acurateții și scăderea loss-ului este crearea unui model simplu și adăugarea unor modificări mici pentru fiecare etapă (astfel, se pot observa plusurile și minusurile modelului ales);
  - Nu am reușit să găsesc o modalitate în care să antrenez modelul folosind imaginile la scara originală aplicând și normalizare. Pe platformă, RAM-ul alocat (13GB pentru configurația cu GPU) nu este suficient pentru stocarea acestora (împreună cu varianta normalizată). Local, epocile durau mult mai mult, această variantă nefiind una acceptabilă.

- **CNN v1**

  Primul model este de forma:

Layer	**param_1**	**param_2**
Conv2D	filters=128	kernel_size=(3,3)
MaxPooling2D		
Conv2D	filters=128	kernel_size=(3,3)
MaxPooling2D		
Conv2D	filters=256	kernel_size=(3,3)
MaxPooling2D		
Conv2D	filters=256	kernel_size=(3,3)
MaxPooling2D		
Conv2D	filters=512	kernel_size=(3,3)
MaxPooling2D		
Conv2D	filters=512	kernel_size=(3,3)
MaxPooling2D		
Flatten		
Dense	filters=1024	
Dense	filters=1024	
Dense	filters=1	

Utilizând acest model, împreună cu o funcție pentru early stopping (oprește antrenarea modelului în momentul în care parametrul urmărit nu se îmbunătățește după un număr setat de epoci), am reușit să ating un scor de 0.55813 pe Kaggle.

Am monitorizat loss-ul pentru 3 epoci, modelul având ca parametru 150 de epoci.

class	**precision**	**recall**	**f1-score**
0	0.91	0.96	0.93
1	0.63	0.39	0.48

Accuracy (0.88)	**precision**	**recall**	**f1-score**
macro avg	0.77	0.68	0.71
weighted avg	0.87	0.88	0.87

	P0	**P1**
T0	1659	65
T1	167	109

- **CNN v2**

  Următoarea variantă a modelului folosește o diferită metodă de citire (utilizând librăria pillow), transformând imaginile în grayscale și făcându-le resize la 128x128 pixeli. Astfel, reducem imaginile de la 3 dimensiuni (RGB) la o singură dimensiune. De asemenea, crește performanța și scade timpul de de rulare și de antrenare al modelului. În dataset-ul prezentat, consider că 3 canale (RGB) nu aduc un plus în antrenarea modelului, întrucât avem doar 2 clase, iar imaginile reprezintă tomografii. Se pot observa anomaliile fără a fi necesare 3 canale de culoare. De asemenea, imaginile au fost normalizate folosind împărțirea la 255. Astfel, din intervalul de valori [0, 255], formăm intervalul [0, 1]. Modelul poate învăța mai ușor folosind aceste valori din noul interval.

  De asemenea, am adăugat un dropout după fiecare 2 layere, cu o valoare de 0.25, pentru a preveni overfitting-ul. Astfel, la fiecare iterație, 25% de neuroni din layer-ul precedent vor fi dezactivați temporar. În plus, am încercat să scad numărul de filtre din fiecare layer și să adaug un layer în plus.

  Pentru resize-ul imaginilor am folosit resampling de tip “bicubic”, care preia o matrice de 4x4 de pixeli, le face smoothing și creează un nou pixel. Am incercat si resampling de tip “lanczos”, doar că a performat mai slab. De asemnea, pentru că am făcut imaginile de tip grayscale, a

fost necesar să adaug o dimensiune în plus, shape-ul obținut fiind de forma (height, width, 1).

Parametrul monitorizat în funcția de early stopping a devenit “val_loss”, deoarece reprezintă o modalitate mai bună de determinare a stării în care se află modelul (dacă loss-ul pe setul de validare nu scade după 5 epoci, antrenarea modelului este oprită).

Straturile sunt următoarele:

Layer	**param_1**	**param_2**
Conv2D	filters=32	kernel_size=(3,3)
BatchNormalization		
Conv2D	filters=32	kernel_size=(3,3)
BatchNormalization		
MaxPooling2D		
Dropout	rate=0.25	
Conv2D	filters=64	kernel_size=(3,3)
BatchNormalization		
Conv2D	filters=64	kernel_size=(3,3)
BatchNormalization		
MaxPooling2D		
Dropout	rate=0.25	
Conv2D	filters=128	kernel_size=(3,3)

BatchNormalization		
Conv2D	filters=128	kernel_size=(3,3)
BatchNormalization		
MaxPooling2D		
Dropout	rate=0.25	
Conv2D	filters=128	kernel_size=(3,3)
BatchNormalization		
Conv2D	filters=128	kernel_size=(3,3)
BatchNormalization		
MaxPooling2D		
Dropout	rate=0.25	
Flatten		
Dense	filters=512	
Dense	filters=512	
Dense	filters=1	

class	**precision**	**recall**	**f1-score**
0	0.93	0.96	0.94
1	0.68	0.52	0.59

Accuracy (0.90)	precision	recall	f1-score
macro avg	0.8	0.74	0.77
weighted avg	0.89	0.9	0.89

	P0	P1
T0	1657	67
T1	132	144

- **CNN v3**

  Pentru varianta a 3-a a modelului, am făcut câteva modificări. Am început să folosesc clasa “ImageDataGenerator” din “keras”, pentru a augmenta imaginile de antrenament. De exemplu, singurul parametru folosit pentru acest model a fost rotation_range=30, care indică faptul că imaginile de training pot fi rotite cu până la 30 de grade la stânga sau la dreapta, această valoare fiind aleasă aleator (la fel se întâmplă pentru toți parametrii acestei clasa).

  De asemenea, numărul de filtre al ultimelor 2 layere convoluționale a fost modificat din 128 în 256, iar cele 2

layere de tip dense au primit 256 de filtre în loc de 512. În plus, am adăugat încă un layer de dropout, cu o valoare de 0.4, înainte de layer-ul de output. Acesta indică faptul că 40% din neuroni vor fi dezactivați temporar, ceea ce poate conduce la o generalizare mai bună a modelului, trebuind să se bazeze pe neuronii care încă au informație pentru antrenare.

Pentru toate modelele care urmează, parametrul batch_size din funcția “fit” a fost modificat din 64 în 32, ceea ce înseamnă că modelul va procesa 32 de imagini în același timp, în loc de 64. Un batch size mai mare implică faptul că parametrii modelului (weights și biases) vor fi actualizați mai rar, în timp ce, pentru 32, vor avea parte de o actualizare mai frecventă.

O ultimă modificare adusă modelului este creșterea parametrului patience din funcția de early stopping de la 5 la 7, pentru a îi oferi modelului o șansă mai mare să își ajusteze parametrii într-o direcție mai bună, neoprindu-l prematur.

Acest model a obținut un scor de 0.6 pe Kaggle.

Layer	**param_1**	**param_2**
Conv2D	filters=32	kernel_size=(3,3)
BatchNormalization		
Conv2D	filters=32	kernel_size=(3,3)
BatchNormalization		
MaxPooling2D		

Dropout	rate=0.25	
Conv2D	filters=64	kernel_size=(3,3)
BatchNormalization		
Conv2D	filters=64	kernel_size=(3,3)
BatchNormalization		
MaxPooling2D		
Dropout	rate=0.25	
Conv2D	filters=128	kernel_size=(3,3)
BatchNormalization		
Conv2D	filters=128	kernel_size=(3,3)
BatchNormalization		
MaxPooling2D		
Dropout	rate=0.25	
Conv2D	filters=256	kernel_size=(3,3)
BatchNormalization		
Conv2D	filters=256	kernel_size=(3,3)
BatchNormalization		
MaxPooling2D		
Dropout	rate=0.25	
Flatten		
Dense	filters=256	

Dense	filters=256	
Dropout	rate=0.4	
Dense	filters=1	

class	**precision**	**recall**	**f1-score**
0	0.92	0.99	0.95
1	0.83	0.46	0.59

Accuracy (0.91)	**precision**	**recall**	**f1-score**
macro avg	0.88	0.72	0.77
weighted avg	0.91	0.91	0.9

	P0	**P1**
T0	1699	25
T1	150	126

- **CNN v4**

A patra variantă a modelului implică modificări majore în clasa ImageDataGenerator. A fost redus numărul maxim de grade din rotația aleatorie a imaginilor, dar au fost adăugați noi parametrii. Parametrii width_shift_range și height_shift_range permit modelului să shift-eze imaginile de antrenament cu până la 20% pe axa X, respectiv Y, zoom_range cu o valoare de 0.2 poate conduce la un zoom de +/-20% asupra imaginilor, iar horizontal_flip este de la sine înțeles.

Acești parametrii au fost adăugați pentru a preveni overfitting-ul și pentru a îi oferi modelului acces la imagini “noi”, ceea ce conduce la o generalizare mai bună a feature-urilor. Mai jos se află tabelul cu noile valori din clasa ImageDataGenerator.

Parameter	**Value**
rotation_range	15
width_shift_range	0.2
height_shift_range	0.2
zoom_range	0.2
horizontal_flip	True

Am făcut câteva modificări și la numărul de filtre de la unele straturi. Layerele au acum câte 128, 256, 512, respectiv 1024. A fost dublat numărul de filtre și din straturile dense.

Am ales să cresc aceste valori pentru a îi permite modelului să învețe trăsături mai complexe. De asemenea, parametrul patience din funcția de early stopping a fost crescut la 15, iar numărul de epoci la 150, pentru a îi da posibilitatea modelului să evolueze pentru mai multe epoci.

O modificare nouă adusă acestei variante este introducerea parametrului class_weight, care are valorile 1 pentru 0 și 2.5 pentru 1, ceea ce înseamnă că va fi pus mai mult accent pe clasa din urmă. Această variantă a modelului a obținut un scor de 0.67782 pe Kaggle. Mai jos este prezentat tabelul cu straturile folosite.

Layer	**param_1**	**param_2**
Conv2D	filters=128	kernel_size=(3,3)
BatchNormalization		
Conv2D	filters=128	kernel_size=(3,3)
BatchNormalization		
MaxPooling2D		
Dropout	rate=0.25	
Conv2D	filters=256	kernel_size=(3,3)
BatchNormalization		
Conv2D	filters=256	kernel_size=(3,3)
BatchNormalization		

MaxPooling2D		
Dropout	rate=0.25	
Conv2D	filters=512	kernel_size=(3,3)
BatchNormalization		
Conv2D	filters=512	kernel_size=(3,3)
BatchNormalization		
MaxPooling2D		
Dropout	rate=0.25	
Conv2D	filters=1024	kernel_size=(3,3)
BatchNormalization		
Conv2D	filters=1024	kernel_size=(3,3)
BatchNormalization		
MaxPooling2D		
Dropout	rate=0.25	
Flatten		
Dense	filters=512	
Dense	filters=512	
Dropout	rate=0.4	
Dense	filters=1	

class	precision	recall	f1-score
0	0.9	0.99	0.94
1	0.8	0.31	0.45

Accuracy (0.89)	precision	recall	f1-score
macro avg	0.85	0.65	0.69
weighted avg	0.89	0.89	0.87

	P0	P1
T0	1702	22
T1	190	86

- **CNN v5**

A 5-a variantă a modelului prezintă modificări asupra clasei ImageDataGenerator. Am observat puțin underfitting în modelul precedent, așa că am decis să scad valorile. Aceste modificări minore au adus o creștere considerabilă a scorului pe Kaggle, acesta ajungând la 0.70204.

Parameter	Value
rotation_range	10
width_shift_range	0.1
height_shift_range	0.1
zoom_range	0.1
horizontal_flip	True

class	precision	recall	f1-score
0	0.94	0.96	0.95
1	0.7	0.61	0.65

Accuracy (0.91)	precision	recall	f1-score

macro avg	0.82	0.78	0.8
weighted avg	0.91	0.91	0.91

	P0	**P1**
T0	1653	71
T1	200	76

- **CNN v6**

  Următoarea iterație a modelului prezintă un număr considerabil de modificări. O primă modificare o reprezintă, din nou, schimbarea valorilor parametrilor din clasa ImageDataGenerator. Acest fine tuning este realizat pentru găsirea celei mai bune combinații de parametri care să obțină un scor cat mai mare. Parametrii sunt prezentați mai jos.

Parameter	**Value**
rotation_range	12
width_shift_range	0.2

height_shift_range	0.2
zoom_range	0.12
horizontal_flip	True

O altă modificare adusă modelului a fost găsirea unei valori cât mai mari pentru numărul de pixeli al imaginilor. Am încercat mai multe valori (pornind de la cea de bază, 224) până am ajuns la una care nu cauza resetarea notebook-ului de pe Kaggle (din cauza memoriei RAM). Această valoare este 188x188 pixeli.

Următorul lucru pe care am încercat să-l fac este reducerea timpului de antrenare al modelului. Am decis să scad, din nou, numărul de filtre din fiecare layer. Noua configurație include valori de la 32 la 512. De asemenea, după cele 2 straturi Dense am adăugat câte un layer de BatchNormalization. Cu ajutorul acestora, se stabilizează procesul de antrenare al modelului și crește viteza de convergență la soluția optimă.

În plus, am decis să revin la o variantă a modelului fără a implementa class weights, deoarece, dacă acestea nu sunt alese bine, pot influența negativ evoluția modelului. O ultimă modificare este creșterea parametrului patience, din funcția de early stopping, la 20, pentru a îi oferi și mai mult timp modelului să evolueze într-o direcție pozitivă.

Această variantă a modelului a obținut cel mai bun scor pe Kaggle, respectiv 0.73553.

Layer	param_1	param_2
Conv2D	filters=32	kernel_size=(3,3)
BatchNormalization		
Conv2D	filters=32	kernel_size=(3,3)
BatchNormalization		
MaxPooling2D		
Dropout	rate=0.25	
Conv2D	filters=64	kernel_size=(3,3)
BatchNormalization		
Conv2D	filters=64	kernel_size=(3,3)
BatchNormalization		
MaxPooling2D		
Dropout	rate=0.25	
Conv2D	filters=128	kernel_size=(3,3)
BatchNormalization		
Conv2D	filters=128	kernel_size=(3,3)
BatchNormalization		
MaxPooling2D		
Dropout	rate=0.25	
Conv2D	filters=256	kernel_size=(3,3)
BatchNormalization		

Conv2D	filters=256	kernel_size=(3,3)
BatchNormalization		
MaxPooling2D		
Dropout	rate=0.25	
Flatten		
Dense	filters=512	
Dense	filters=512	
Dropout	rate=0.4	
Dense	filters=1	

class	**precision**	**recall**	**f1-score**
0	0.94	0.99	0.96
1	0.88	0.61	0.72

Accuracy (0.93)	**precision**	**recall**	**f1-score**
macro avg	0.91	0.8	0.84
weighted avg	0.93	0.93	0.93

	P0	P1
T0	1702	22
T1	109	167

- **Concluzii**

  După analiza celor 6 modele, am tras câteva concluzii. Printre acestea, se numără:

  - Folosirea clasei ImageDataGenerator poate îmbunătăți destul de mult acuratețea modelului, dacă aceasta este folosită într-un mod corect, care nu produce overfitting (augmentarea imaginilor prea mult poate duce la un grad de generalizare foarte slab, modelul nereușind să învețe trăsăturile necesare, ci doar să rețină imaginile);
  - Layer-ul Dropout influențează foarte mult evoluția modelului. Am observat că o schimbare de 10-15% poate înrăutăți destul de mult acuratețea pe care o prezintă rețeaua neuronală;
  - Am încercat să adaug și un layer RandomContrast modelului, care să aleagă un contrast aleator pentru imaginile de antrenare. Adăugarea acestuia la ultima variantă (CNN v6) creștea timpul de antrenare pe o singură epocă la o valoare destul de mare, așa că nu am continuat să-l antrenez. Presupun că o rețea neuronală mai complexă, cu o adâncime mai mare, o

creștere a parametrilor din clasa ImageDataGenerator, a valorilor acestora și adăugarea unui layer RandomContrast ar putea îmbunătăți performanțele atinse de model;
- Chiar dacă pentru un model classification report-ul și confusion matrix-ul sunt mai bune decât pentru altul, nu înseamnă neapărat că produce o clasificare mai bună a imaginilor de test.

- **Bibliografie**
  - keras.io
  - tensorflow.org
  - scikit-learn.org