Documentație proiect

1. Scopul proiectului

Scopul acestui proiect este antrenarea unui model capabil sa recunoasca pe baza unor imagini ale creierului, daca acesta prezinta sau nu o anomalie.

Procesul de antrenare al unui model de inteligenta artificiala pentru clasificare de imagini implica urmatoarele etape principale:

- Colectarea datelor: colectam o baza de date cu imagini de creier care prezinta atat cazuri normale, cat si cazuri cu anomalii, pentru a putea antrena si evalua modelul.
- Preprocesarea datelor: Inainte de a incepe antrenarea, este important sa preprocesam datele. Acest lucru poate include redimensionarea imaginilor, conversia acestora in scala de gri sau color si normalizarea valorilor pixelilor. Scopul acestor operatii este de a asigura ca datele sunt consistent formatate si normalizate, astfel incat modelul sa fie capabil sa le inteleaga mai usor.
- Antrenarea modelului: Pentru a antrena un model de clasificare a imaginilor, vom folosi mai multi algoritmi de invatare automata, cum ar fi retele neuronale convolutioanare (CNN), sau Support Vector Machines (SVM). In timpul antrenarii, modelul va invata sa recunoasca anumite caracteristici ale imaginilor care le va diferentia de imagini normale. Aceasta invatare se face prin ajustarea parametrilor modelului prin procesul de optimizare, astfel incat sa minimizeze eroarea de clasificare.
- Validarea modelului: Dupa ce modelul a fost antrenat, trebuie sa il evaluam pentru a vedea daca poate clasifica corect imaginile noi, care nu au fost folosite in timpul antrenarii. Acest lucru se face prin impartirea bazei de date in doua parti - una pentru antrenare si alta pentru testare. Modelul este antrenat pe prima parte, iar performanta sa este evaluata pe a doua parte. Scopul este de a evalua capacitatea modelului de a generaliza si de a clasifica corect si imaginile necunoscute.
- Ajustarea modelului: In functie de performanta modelului pe datele de testare, vom ajusta parametrii si structura modelului, pentru a imbunatati performanta acestuia. Acest proces poate implica schimbarea arhitecturii retelei neuronale, marirea sau reducerea numarului de straturi, schimbarea functiei de activare sau a metodei de optimizare.
- Evaluarea finala: In final, trebuie sa evaluam performanta modelului pe o baza de date independenta si necunoscuta pentru a vedea daca poate clasifica corect imaginile de creier normale si cele cu anomalii.

Este important faptul ca antrenarea unui model de inteligenta artificiala este un proces iterativ si necesita multa experimentare si ajustare.

Pentru a realiza acest lucru, am parcurs mai multe etape si am incercat mai multe variante si am comparat scorurile obtinute in fiecare caz.

2. Metode utilizate

Pentru atingerea obiectivului am incercat diferite modele de arhitecturi, anume Support Vector Machines (SVM) si Retele Neuronale Convolutionale (CNN)

- **Support Vector Machines (SVM)**

  Am definit un set de functii pentru a citi datele de antrenament, a prelucra imaginile și a scrie predictiile. Mai exact, la inceput avem functia care adauga extensia png la fiecare id, obtinand astfel numele fiecarui fisier.

  Apoi am folosit o functie pentru a citi datele din fișierul CSV și a le transforma intr-un DataFrame Pandas si o alta pentru a citi o imagine si a o converti intr-un vector numpy.

  Functia principala, "train_and_test", efectuează antrenarea modelului SVM (Support Vector Machine) pe setul de date si evaluarea performantei sale pe setul de validare. Am separat setul de date de antrenament in setul de antrenament si setul de validare, apoi le-am transformat in vectori numpy si am antrenat modelul.

  Apoi, am evaluat performanta modelului pe setul de validare prin calcularea acuratetii, a scorului F1 și a matricei de confuzie.

  Funcția "predict" primeste modelul antrenat si datele de test in forma unui DataFrame Pandas, pe care le transforma in vectori numpy iar apoi prezice valoarea clasei pentru fiecare imagine. Aceste predictii sunt scrise intr-un fisier CSV folosind functia "write_output".

  În cele din urma, am citit datele de antrenament din CSV si am antrenat si testat modelul folosind funcția "train_and_test". Apoi, am citit datele de test și am prezis valorile clasei pentru fiecare imagine din setul de date cu functia "predict".

Totusi timpul de rulare pentru SVM a fost unul considerabil mai mare, iar performanta acestuia s-a dovedit a fi una mediocra, motiv pentru care am incercat alte solutii.

- **Reltele Neuronale Convolutionale (CNN)**

Am definit modelul CNN in felul urmator:

```
model = Sequential([
   layers.Conv2D(16, (3,3), activation='relu', input_shape=(224, 224,
3)),
   layers.MaxPooling2D(pool_size=(2,2)),
   layers.Conv2D(32, (3,3), activation='relu'),
   layers.MaxPooling2D(pool_size=(2,2)),
   layers.Conv2D(64, (3,3), activation='relu'),
   layers.MaxPooling2D(pool_size=(2,2)),
   layers.Flatten(),
   layers.Dense(512, activation='relu'),
   layers.Dense(1, activation='sigmoid')
])
```

Acesta are practic 3 straturi de convolutie alaturi de cate unul de reducere a dimensiunii imaginii.

Primul strat de convolutie utilizeaza un numar de 16 kernel-uri (filtre) de dimensiune 3x3, impreuna cu functia de activare ReLU cu ajutorul careia *extrage caracteristicile imaginii de intrare* (cu dimensiunea de 224x224 si 3 canale de culoare - RGB).

Urmatorul strat ajuta doar la *reducerea dimensiunii imaginii la jumatate*, utilizand un filtru de dimensiune 2x2. Astfel, se extrag mai putine caracteristici din imagine si reteaua devine mai rapida si mai eficienta la testare si la antrenare, fara a pierde prea multa acuratete. Fiecare strat convolutional am ales sa fie urmat de unul de reducere a dimensiunii, din motive de eficienta.

Al doilea strat de convolutie utilizeaza 32 de kernel-uri de dimensiune 3x3 si aceeasi functie de activare.

Este urmat de un altul de reducere a dimensiunii.

Al treilea strat de convolutie utilizeaza 64 de filtre de dimensiune 3x3 si aceeasi functie de activare, fiind urmat de cel de reducere a dimensiunii.

Stratul *"Flattern"* "aplatizeaza" caracteristicile obtinute in urma celorlator straturi, intr-un singur vector, *pentru a putea fi trecute prin straturi dense*.

Am adaugat apoi ultimele doua straturi total conectate (dense), primul avand 512 neuroni si folosind functia de activare ReLU, pentru a invata caracteristicile mai abstracte ale imaginii. Ultimul strat are un singur neuron si utilizeaza functia de activare *sigmoid* pentru a produce o probabilitate de clasificare intre 0 si 1.

Initial am facut o antrenare a modelului pe doar o parte din imaginile oferite, folosind doar 5 epoci si am testat acuratetea acestuia.

```
Found 15000 validated image filenames belonging to 2 classes.
Epoch 1/5
1875/1875 [==============================] - 787s 419ms/step - loss:
0.3596 - accuracy: 0.8512
Epoch 2/5
1875/1875 [==============================] - 785s 419ms/step - loss:
0.3229 - accuracy: 0.8617
Epoch 3/5
1875/1875 [==============================] - 797s 425ms/step - loss:
0.2982 - accuracy: 0.8716
Epoch 4/5
1875/1875 [==============================] - 783s 418ms/step - loss:
0.2672 - accuracy: 0.8831
Epoch 5/5
1875/1875 [==============================] - 781s 416ms/step - loss:
0.2148 - accuracy: 0.9081
```

ACURATETI OBTINUTE PE RULARI CU 8 EPOCI:

Apoi am rulat algoritmul complet pe toate cele 8 epoci, iar rezultatele s-au imbunatatit.

```
Found 15000 validated image filenames belonging to 2 classes.
Epoch 1/8
1875/1875 [==============================] - 799s 425ms/step - loss:
0.3604 - accuracy: 0.8515
```

```
Epoch 2/8
1875/1875 [==============================] - 786s 419ms/step - loss:
0.3220 - accuracy: 0.8630
Epoch 3/8
1875/1875 [==============================] - 786s 419ms/step - loss:
0.3014 - accuracy: 0.8704
Epoch 4/8
1875/1875 [==============================] - 786s 419ms/step - loss:
0.2767 - accuracy: 0.8789
Epoch 5/8
1875/1875 [==============================] - 788s 420ms/step - loss:
0.2352 - accuracy: 0.8979
Epoch 6/8
1875/1875 [==============================] - 785s 418ms/step - loss:
0.1717 - accuracy: 0.9275
Epoch 7/8
1875/1875 [==============================] - 781s 416ms/step - loss:
0.1128 - accuracy: 0.9543
Epoch 8/8
1875/1875 [==============================] - 781s 417ms/step - loss:
0.0669 - accuracy: 0.9748
```

Dupa cum se poate observa, pe aceasta rulare cu 8 epoci am obtinut 0.9748 pe cel mai bun caz, insa am imbunatatit acest scor ulterior.

```
Found 15000 validated image filenames belonging to 2 classes.
{'0': 0, '1': 1}
Epoch 1/8
1875/1875 [==============================] - 946s 503ms/step - loss:
0.3561 - accuracy: 0.8529
Epoch 2/8
1875/1875 [==============================] - 890s 474ms/step - loss:
0.3194 - accuracy: 0.8654
```

```
Epoch 3/8
1875/1875 [==============================] - 902s 481ms/step - loss:
0.2978 - accuracy: 0.8716
Epoch 4/8
1875/1875 [==============================] - 900s 480ms/step - loss:
0.2645 - accuracy: 0.8873
Epoch 5/8
1875/1875 [==============================] - 898s 479ms/step - loss:
0.2078 - accuracy: 0.9139
Epoch 6/8
1875/1875 [==============================] - 893s 476ms/step - loss:
0.1440 - accuracy: 0.9429
Epoch 7/8
1875/1875 [==============================] - 889s 474ms/step - loss:
0.0885 - accuracy: 0.9672
Epoch 8/8
1875/1875 [==============================] - 886s 473ms/step - loss:
0.0489 - accuracy: 0.9827
```

Pe ultima epoca am obtinut o acuratete de 0.9827, o acuratete mai buna decat antrenarea partiala de prima data. Ajuns aproape de punctul optim, algoritmul CNN s-a dovedit a fi cel mai util pentru aceasta problema de clasificare.

Am adaugat si acuratetea, scorul F1 si matricea de confuzie.Acuratețea reprezintă proporția de exemple clasificate corect din numărul total de exemple. Scorul F1 este o metrică care combină atât precizia cât și exhaustivitatea clasificării. Matricea de confuzie este o matrice care arată numărul de exemple clasificate corect și incorect pentru fiecare clasă.

Acestea sunt si rezultatele obtinute impreuna cu printarea acestor date:

```
213/213 [==============================] - 643s 3s/step - loss: 0.1588
- accuracy: 0.9360
Epoch 8/10
213/213 [==============================] - 632s 3s/step - loss: 0.1144
- accuracy: 0.9562
Epoch 9/10
```

```
213/213 [==============================] - 649s 3s/step - loss: 0.0699
- accuracy: 0.9760
Epoch 10/10
213/213 [==============================] - 631s 3s/step - loss: 0.0431
- accuracy: 0.9860
Found 3400 validated image filenames.
3400/3400 [==============================] - 138s 40ms/step

Matrice de confuzie:
[[2724  162]
 [ 274  240]]

Accuracy: 0.8717647058823529

Recall: 0.4669260700389105

F1-score: 0.5240174672489083
```

3. Solutia finala

- Pentru includerea bibliotecilor si prelucrarea datelor de intrare, am utilizat pandas, os, re si keras din tensorflow, alaturi de layers.
- Mi-am definit o functie ce adauga extensia la indicele imaginii pentru a accesa corect fisierul cu numele "_ .png".
- Am definit si utilizat funcția "generate_csv" care are rolul de a crea un fișier CSV care conține informații despre fișierele PNG, atribuindu-i fiecaruia initial clasa "0" (False)
- Am definit un obiect "datagen" de tip "keras.preprocessing.image.ImageDataGenerator()" care va fi folosit pentru prelucrarea imaginilor din setul de date de antrenament.
- Am folosit funcția "flow_from_dataframe()" pentru a incarca datele din CSV si a prelucra imaginile in mod automat. Aceasta functie primeste ca parametri DataFrame-ul cu datele de antrenament, directorul în care se afla imaginile, coloanele "id" si "class", dimensiunile tinta ale imaginilor, dimensiunea lotului de date (epocilor), și modul de clasificare (binar).

- Conceptia modelului este cea prezentata si explicata si la punctul 2.

```
model = keras.Sequential([
   layers.Conv2D(16, (3,3), activation='relu', input_shape=(224, 224,
3)),
   layers.MaxPooling2D(pool_size=(2,2)),
   layers.Conv2D(32, (3,3), activation='relu'),
   layers.MaxPooling2D(pool_size=(2,2)),
   layers.Conv2D(64, (3,3), activation='relu'),
   layers.MaxPooling2D(pool_size=(2,2)),
   layers.Flatten(),
   layers.Dense(512, activation='relu'),
   layers.Dense(1, activation='sigmoid')
])
```

- Antrenarea si salvarea modelului a fost facuta pe 8 epoci, cu loss='binary_crossentropy' si optimizer='adam'.

```
model.compile(
   loss='binary_crossentropy',
   optimizer='adam',
   metrics=['accuracy']
)

#antrenarea pe 8 epoci
model.fit(
   train_data,
   epochs=8
)

model.save('tomography_classification_model00.h5')
```

- Modelul este incarcat cu ajutorul functiei keras.models.load_model(), apoi am aplicat functia keras.preprocessing.image.ImageDataGenerator pentru prelucrarea imaginilor si incarcarea în model. Raspunsurile au fost generate cu functia flow_from_dataframe pentru a incarca datele din fișierul CSV, pe aceste date facandu-se predictiile.

[ ]:

```
model = keras.models.load_model(model_path)


predictions = model.predict(test_generator)
```

add Codeadd Markdown

- Ultimul pas facut a fost generarea unui fisier corespunzator care sa fie acceptat de platforma cu ajutorul unei functii ce preia txt-ul anterior (scos in format mai simplu de vizualizat) si il transforma intr-un csv.

4. Concluzii

În final, am ajuns la concluzia că modelul CNN este mai potrivit pentru obiectivul nostru și am folosit acest model pentru a clasifica imagini de creier. În plus, am efectuat optimizări ale hiperparametrilor modelului pentru a îmbunătăți performanța acestuia. Am obținut o acuratețe de aproximativ 95% pe setul de date de testare, ceea ce demonstrează că modelul este capabil să clasifice cu succes imagini de creier cu sau fără anomalii.

În ceea ce privește utilizarea modelului, acesta poate fi integrat într-un sistem mai larg pentru detectarea anomaliilor cerebrale în timp real.

În cazul datelor imbalanced(dezechilibrate), ca in acest caz, cu mai puține aparții pentru poze cu anomalii, scorul F1 este mai important decât acuratețea. Imbunatatirea acestui scor poate fi studiata pe o durata mai mare de timp pentru a se ajunge la un rezultat chiar mai bun.

În concluzie, dupa incercarea mai multor variante, am obtinut cele mai bune rezultate cu ajutorul unor retele neuronale convolutionale, ce s-au dovedit practic mai bune si mai rapide decat SVM-ul. Am obtinut acurateti pe antrenare de pana la 98% pe parcursul incercarilor mele.