CT Scan Documentation

RADU GEORGE-DANIEL, GRUPA 311

1 Datele proiectului

1.1 Cerinta

Participantii au primit mai multe fisiere de tip text continand numele imaginilor impreuna cu o eticheta si seturile de imagini.
Se cere sa se antreneze datele primite pe un model la alegere si sa se faca predictie pe setul de testare in vederea clasificarii imaginilor cu unul dintre label-urile date.

1.2 Citirea datelor

Pentru fiecare set de date am creat o clasa cu ajutorul careia vom pune inputurile intr-un DataLoader, pe care il vom folosi ulterior pentru extragere la antrenare, evaluare si testare.

```
date_workout = pd.read_csv("train.txt", header=None, names=["cale", "eticheta"])

workout_nume = date_workout.loc[:, "cale"].tolist()
workout_etichete = date_workout.loc[:, "eticheta"].tolist()
workout_nume = [os.path.join('train', i) for i in workout_nume]

date_test = pd.read_csv("test.txt", header=None, names=["cale", "eticheta"])

test_nume = date_test.loc[:, "cale"].tolist()
test_etichete = date_test.loc[:, "eticheta"].tolist()
test_nume = [os.path.join('test', j) for j in test_nume]

date_validation = pd.read_csv("validation.txt", header=None, names=["nume", "eticheta"])

validation_nume = date_validation.loc[:, "nume"].tolist()
validation_etichete = date_validation.loc[:, "eticheta"].tolist()
validation_nume = [os.path.join('validation', k) for k in validation_nume]
```

Figure 1: Citirea si pregatirea datelor

2 Convolutional Network

Abordarea aleasa de mine a fost folosirea unui CNN (Convolutional Neural Networks) deoarece este o metoda foarte populara atunci cand vine vorba despre analizarea imaginilor. Cu alte cuvinte, CNN este un algoritm de deep learning capabil sa recunoasca tipare.

2.1 Cum functioneaza?

Sa presupunem ca vrem sa recunoastem o imagine. Calculatorul va vedea aceasta imagine ca pe un grid ce contine valori rgb (intre 0 si 255).

Peste acest grid se va aplica o "operatie de convolutie", adica, se aplica filtre specifice pentru detectarea tiparului.
Ideea este ca nu se pot aplica din prima filtre complexe, acestea sunt construite pe parcurs ajungandu-se ulterior la un sablon complex (exemplu: pentru a detecta un om intr-o poza se vor aplica intai filtre ce tin de linii verticale /orizontale /curbate, dupa care vor evolua la filtre pentru nas, ochi, gura, iar acestea puse la un loc vor forma tiparul unei persoane).

In mod practic, se aleg toate blocurile de dimensiunea filtrului si se inmultesc valorile intre ele, se aduna, si se calculeaza media. Aceasta medie va reprezenta valoarea unei celule dintr-o harta de caracteristici. Daca avem valori de 1 sau apropiate de 1 rezultate, inseamna ca detectam caracteristici de tipul filtrului aplicat.

Pe aceste "mape" rezultate din convolutie se aplica pooling (de obicei un MAX pooling), care nu face decat sa reduca dimensiuniea si sa creasca perfomanta. Concret, se iau blocuri de dimensiunea parametrilor dati si se extrage cea mai mare valoare dintre ele, formandu-se un alt bloc cu aceste valori.

Dupa ce avem pasii de mai de sus, ultima etapa din extragerea trasaturilor sablonului este sa facem flatten pe ele, adica sa le unim intr-un singur array care va merge mai departe spre clasificare.

2.2 Pregatirea datelor

Dupa cum spuneam mai sus, imaginile trebuie reprezentate sub forma unor griduri cu numere rgb. Pentru asta am folosit o clasa dedicata fiecarui set de imagini si etichete care prelucreaza datele si le returneaza in formatul dorit (folosind torch.tensor()).

```
[13]:
class set_workout(Dataset):
    def __init__(self, workout_nume, workout_etichete):
        self.workout_nume = workout_nume
        self.workout_etichete = workout_etichete

    def __len__(self):
        return len(self.workout_etichete)

    def __getitem__(self, index):
        workout_path = self.workout_nume[index]
        img = Image.open(workout_path)
        img = np.array(img) / 255
        imagine_intoarsa = torch.tensor(img).unsqueeze(0).float()
        eticheta = torch.tensor(self.workout_etichete[index])

        return imagine_intoarsa, eticheta
```

Figure 2: Prelucrare date

2.3 Realizarea modelului

```
import torch.nn as nn
import torch.nn.functional as F
class RaduG(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(in_channels=1, out_channels=40, kernel_size=5, stride = 1)
        self.conv2 = nn.Conv2d(in_channels=40, out_channels=120, kernel_size=5, stride=1)
        self.conv3 = nn.Conv2d(in_channels=120, out_channels=80, kernel_size=5, stride=1)

        self.pool = nn.MaxPool2d(2, 2)
        self.flatten = nn.Flatten()

        self.fc1 = nn.Linear(320, 70)
        self.fc2 = nn.Linear(70, 3)

        self.relu = nn.ReLU()
        self.softmax = nn.Softmax(dim=1)
```

Figure 3: Modelul Convolutional

In primul rand vom creea 3 straturi convolutionale cu metoda Conv2d. Fiecare dintre ele definiste:

- **kernel-size=5** reprezinta dimensiunea filtrului
- **out-channels** reprezinta numarul de filtre aplicate
- **stride** este pasul de mutare al filtrului

Setam pooling-ul cu **MaxPool2(2,2)** pentru a reduce numarul de pixeli din "mapa caracteristicilor" la jumatate.

Flatten converteste "mapa" intr-un array cu o singura dimensiune

Softmax este cel ce ne va spune probabilitatea pentru fiecare predictie si suma sa va fi 1 (aceasta se va aplica ultima).

```
def forward(self, x):

    x = self.conv1(x)
    x = self.relu(x)
    x = self.pool(x)

    x = self.conv2(x)
    x = self.relu(x)
    x = self.pool(x)

    x = self.conv3(x)
    x = self.relu(x)
    x = self.pool(x)

    x = self.flatten(x)
    x = self.fc1(x)
    x = self.relu(x)

    x = self.fc2(x)
    x = self.relu(x)
    x = self.softmax(x)
    return x
```

Figure 4: Model part 2

Relu este functia de activare liniara. Practic, i-a valorile negative si le transforma in 0, iar pe cele pozitiva le lasa asa cum sunt.

Acuratete Kaggle = 0.58222

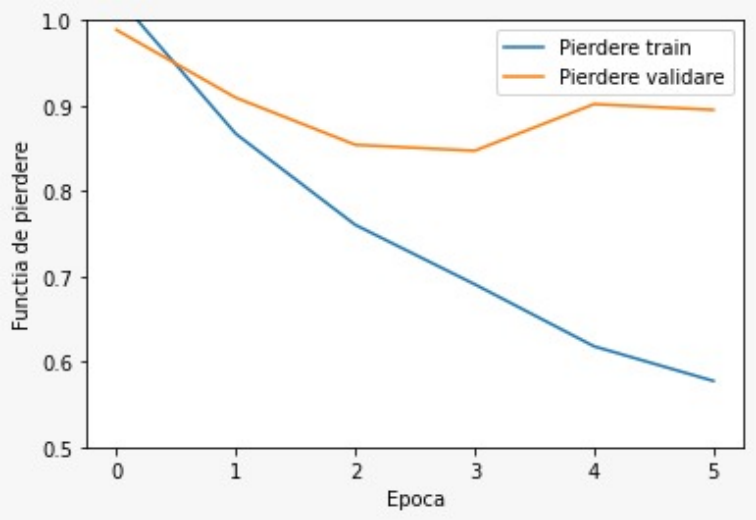

Figure 5: Grafic1

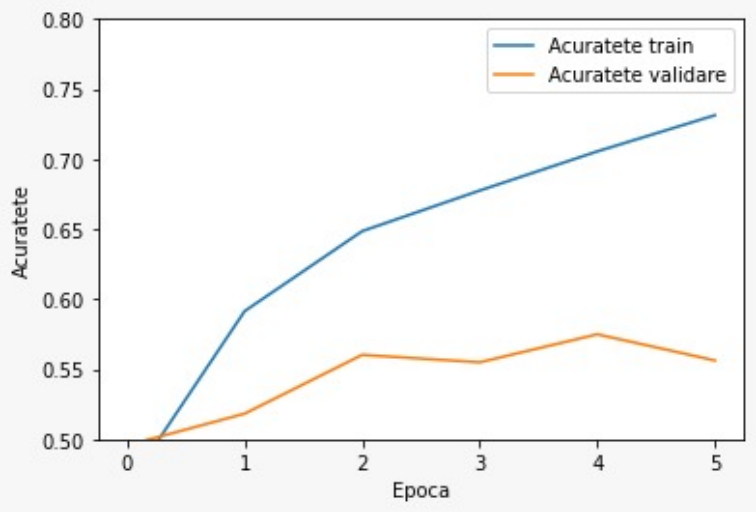

Figure 6: Grafic2

Col1	Lr	Epoci	Acuratete
1	0.00001	6	0.34
2	0.001	6	0.58
3	0.001	10	0.61

2.4 Referinte

- https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-ne
- https://deeplizard.com/learn/video/IKOHHItzukk
- https://cs231n.github.io/convolutional-networks/#conv
- https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-
- https://pytorch.org/tutorials/beginner/blitz/neural_networks_tutorial.html