Opis danych

Zbiór informacji oraz przykładowych kodów, które uznano za przydatne do wykonania projektu.

1. Otwieranie i pobieranie danych

```
! curl -s
https://packagecloud.io/install/repositories/github/git-lfs/script.deb.
sh | sudo bash
! sudo apt-get install git-lfs
! git lfs install
! git clone https://github.com/neheller/kits19

%cd kits19/
! python -m starter_code.get_imaging

from starter_code.utils import load_case
volume, segmentation = load_case("case_00013")
from starter_code.visualize import visualize
visualize("case_00013", 'case13')
```

Rys 1. Struktura pobranych informacji - widok google colab.

load case:

Otrzymano obrazy w formacie NIFTI o kształcie - (num_slices, height, width)

- num_slices - widok osiowy (progress from superior to inferior as the slice index increases)
- height, width - początkowa wartość dla lewej przedniej części pacjenta

datatype - np.float32 i np.uint8

W segmentacji: 0 - wartość tła

1 - wartość nerki

2 - wartość guza

visualize:

obrazy w formacie PNG z wyszczególnionymi nerkami (kolor czerwony) oraz guzem (kolor niebieski)

2. Format NIFTI

NIFTI (Neuroimaging Informatics Technology Initiative) jest to format służący do nauroobrazowania. Zawiera co najmniej trójwymiarowe dane: woksele lub piksele o szerokości, wysokości i głębokości.

Pliki, które mają rozszerzenie *.nii.gz* są skompresowanie (gzip).

Aby załadować zdjęcia można użyć się np. `img = nib.load(example_filename)`

nib - NiBabel https://nipy.org/nibabel/gettingstarted.html

3. Przemieszczanie się między folderami

***sprawdzanie, w którym folderze się znajdujemy, czy istnieje**

Przykłady:

```
import os

os.chdir()
os.path.join()
os.path.exists()
out_path.mkdir()
```

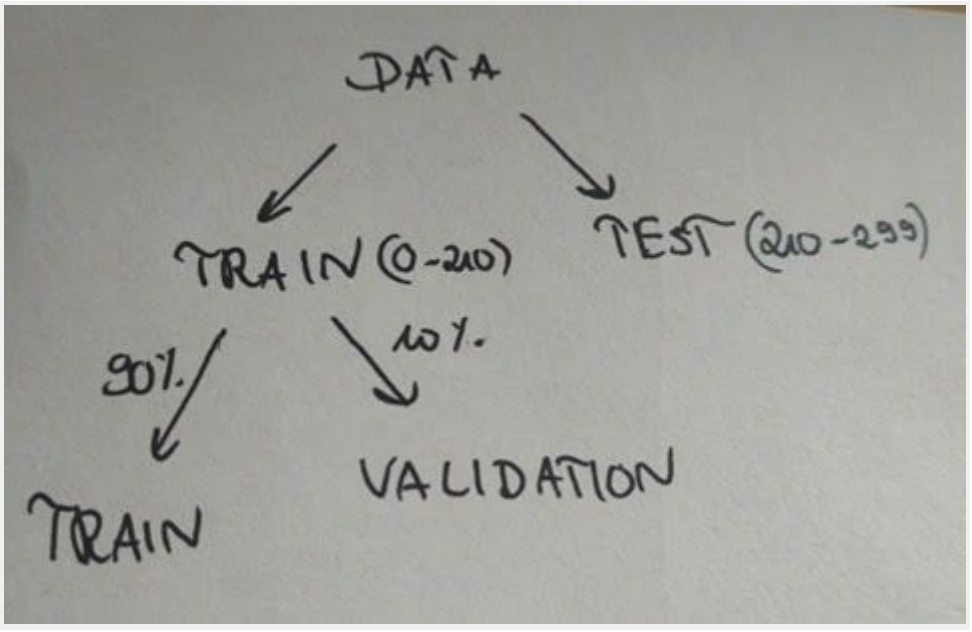
Więcej: https://docs.python.org/3/library/os.path.html

4. Tain and test

Pliki od 210 do 299 należą do przedziału testowego.

Aby odpowiednio je podzielić można wykorzystać:

```
from sklearn.model_selection import train_test_split,cross_val_score
X_train, X_test, y_train, y_test=train_test_split(X, y, test_size=0.1,random_state=0)
```



Rys 2. Drzewo z podziałem na train i test.

tqdm - uzyskujemy pasek postępu

```
from tqdm import tqdm
for i in tqdm(range()):
```

Augmentacja danych obrazu to technika, której można użyć do sztucznego zwiększenia rozmiaru zestawu treningowego, poprzez tworzenie zmodyfikowanych wersji obrazów w zbiorze danych - klasa **ImageDataGenerator**.

flow_from_directory () - Pobiera ścieżkę do katalogu i generuje partie rozszerzonych/znormalizowanych danych.

Przykład

```
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
        '/home/datasets/cifar10/train',
        target_size=(32, 32),
        batch_size=32,
        shuffle=False)
```

Więcej: https://keras.io/api/preprocessing/image/#imagedatagenerator-class

5. Funkcja straty

According to this Keras implementation of Dice Co-eff loss function - the loss is minus of calculated value of dice coefficient.

```
from keras import backend as K

def dice_coef(y_true, y_pred):
    y_true_f = K.flatten(y_true)
    y_pred_f = K.flatten(y_pred)
    intersection = K.sum(y_true_f * y_pred_f)
    return (2. * intersection + smooth) / (K.sum(y_true_f) + K.sum(y_pred_f) + smooth)


def dice_coef_loss(y_true, y_pred):
    return -dice_coef(y_true, y_pred)
```

Można również napisać: *1 - dice_coef*

```
def dice_coef_loss(y_true, y_pred):
    return K.mean(1-dice_coef(y_true, y_pred),axis=-1)
```

Istnieją 4 formy normalizacji danych, wykonywanych podczas treningu BN, IN, LN i GN - zostały one szczegółowo opisane w tym artykule, łącznie ze wzorami.
https://arxiv.org/abs/1809.03783

Artykuł opisujący kroki keras ale dla U-Net, bez 2D :(
https://medium.com/@pallawi.ds/semantic-segmentation-with-u-net-train-and-test-on-your-custom-data-in-keras-39e4f972ec89