WPROWADZENIE DO SIECI NEURONOWYCH

Sprawozdanie z laboratorium

Angelika Nadolska, Anna Wąsik, Kacper Wichowicz

1 Podstawowe informacje

1.1 Wprowadzenie

Głównym celem projektu jest zbudowanie modelu konwolucyjnej sieci neuronowej (CNN), klasyfikującego obrazy MRI na podstawie widoczności guza mózgu. Problem ten został już rozwiązany na platformie Kaggle.
Do wytrenowania modelu użyliśmy architektury VGG-16. Miarą trafności modelu jest *accuracy*, którą definiuje się następująco:

$$accuracy = \frac{C}{T} \cdot 100\%,$$

gdzie C oznacza liczbę poprawnie przewidzianych obrazów, a T całkowitą liczbę testowanych obrazów.

W związku z tym, że nasz zbiór danych był bardzo mały, skorzystaliśmy z tzw. transfer learningu. Także wygenerowaliśmy więcej danych robiąc różne transformacje na obrazkach.

1.2 Konwolucyjne sieci neuronowe (CNN)

Niestety, gdy posiadamy zbyt dużo danych wejściowych istnieje możliwość, że nasza głęboka sieć się "przeuczy". Chodzi tutaj, o to, żeby sieć faktycznie nauczyła się rozpoznawać wilki, a nie sytuacje, że występują one najczęściej na jasnym, zimnym tle (ponieważ w takiej sytuacji sieć rozpoznaje zimę, a nie wilki). Rozwiązaniem tego problemu wydają się być konwolucyjne sieci neuronowe.

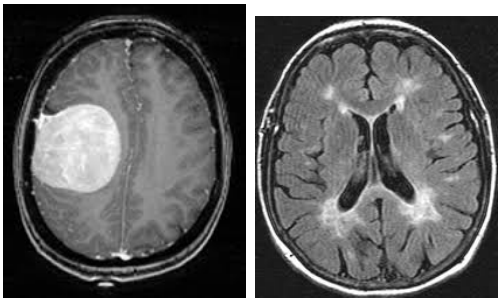
Aby uniknąć przeuczania Głębokich sieci neuronowych konwolucyjne sieci neuronowe badają wydzieloną część obrazu tak samo jak sieć głęboka, jednak skupiają się na wybranej przez nas konkretnej charakterystycznej cesze, odrzucając rzeczy nas nie interesujące.

1.3 Architektura VGG-16

1.4 Transfer learning

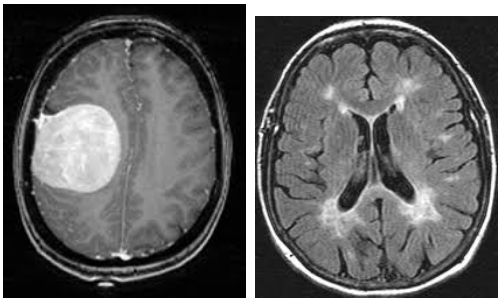
2 Przygotowywanie danych

Nasze dane są już sklasyfikowane na obrazy z nowotworem i bez.



(a) Obraz z nowotworem. (b) Obraz bez nowotworu.

Rysunek 1: Przykładowe dane.



(a) Obraz z nowotworem. (b) Obraz bez nowotworu.

Rysunek 2: Przykładowe dane.

2.1 Podział na zbiory

Pozostaje nam podzielić je na zbiór testowy, treningowy i walidacyjny. Do tego, napisaliśmy program, który losowo wybiera po 5 obrazów do zbioru testowego, następnie 80% reszty losowo do zbioru treningowego, a to co zostało, przydziela do zbioru walidacyjnego.

```
import glob
yes_dir = glob.glob('C:/git/obrazyMRI/data/brain_tumor_dataset/yes/*')
no_dir =glob.glob('C:/git/obrazyMRI/data/brain_tumor_dataset/no/*')

import os

# define the name of the directory to be created
path = ["C:/git/obrazyMRI/data/TEST/YES","C:/git/obrazyMRI/data/TEST/NO"
    , "C:/git/obrazyMRI/data/TRAIN/YES",
        "C:/git/obrazyMRI/data/TRAIN/NO", "C:/git/obrazyMRI/data/VAL/YES"
    , "C:/git/obrazyMRI/data/VAL/NO"]
for i in path:
    try:
        os.makedirs(i)
    except OSError:
```

```
14          print ("Creation of the directory %s failed" % i)
15      else:
16          print ("Successfully created the directory %s" % i)
17
18  from math import *
19  import numpy as np
20  import shutil
21
22  yes_test=np.random.choice(yes_dir, size=5, replace=False)
23  for i in yes_test:
24      shutil.copy(i, "C:/git/obrazyMRI/data/TEST/YES" )
25  yes_dir=list(set(yes_dir)-set(yes_test))
26  yes_train=np.random.choice(yes_dir, size=floor(0.8*len(yes_dir)),
        replace=False)
27  for i in yes_train:
28      shutil.copy(i, "C:/git/obrazyMRI/data/TRAIN/YES" )
29  yes_val=list(set(yes_dir)-set(yes_train))
30  for i in yes_val:
31      shutil.copy(i, "C:/git/obrazyMRI/data/VAL/YES" )
32
33
34  no_test=np.random.choice(no_dir, size=5, replace=False)
35  for i in no_test:
36      shutil.copy(i, "C:/git/obrazyMRI/data/TEST/NO" )
37  no_dir=list(set(no_dir)-set(no_test))
38  no_train=np.random.choice(no_dir, size=floor(0.8*len(no_dir)), replace=
        False)
39  for i in no_train:
40      shutil.copy(i, "C:/git/obrazyMRI/data/TRAIN/NO" )
41  no_val=list(set(no_dir)-set(no_train))
42  for i in no_val:
43      shutil.copy(i, "C:/git/obrazyMRI/data/VAL/NO" )
```

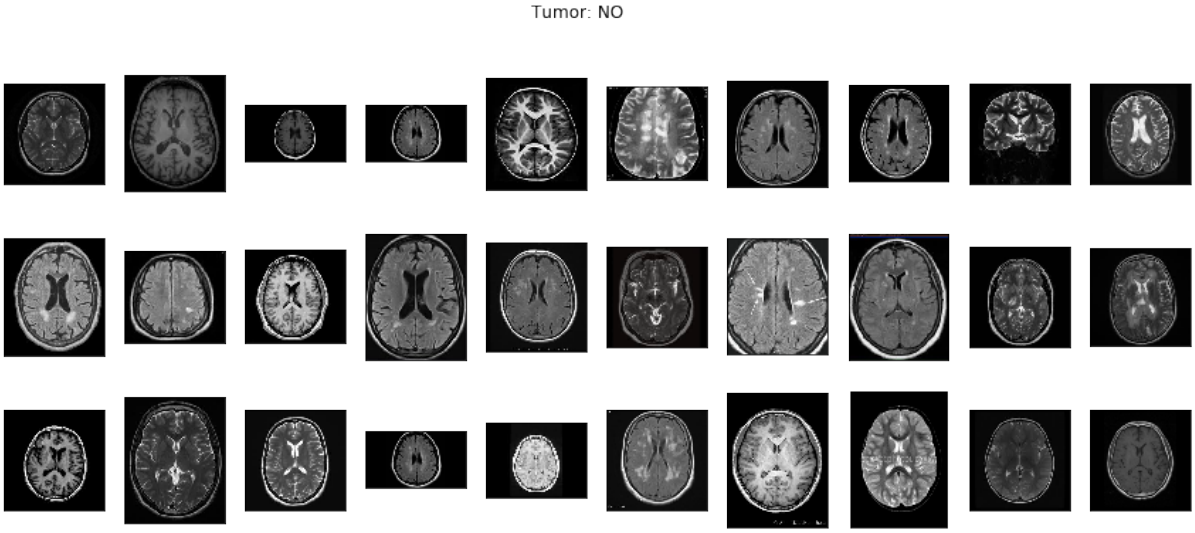
W skrócie - w linijkach 2-3 tworzymy dwie listy napisów, a dokładniej ścieżek do każdego pliku. W zmiennej ‘path‘ znajdują się ścieżki do katalogów, które chcemy stworzyć. Pętla, zaczynająca się od 10-tej linijki tworzy nam foldery przy czym wypisuje, czy udało się je stworzyć, czy nie. W linijce 22-ej losujemy 5 obrazów, następnie pętlą wpisujemy je w folder. W linijce 25-tej odejmujemy obrazki już wylosowane. Reszta kodu jest analogiczna.

Oto przykładowe obrazu dla zbioru treningowego.

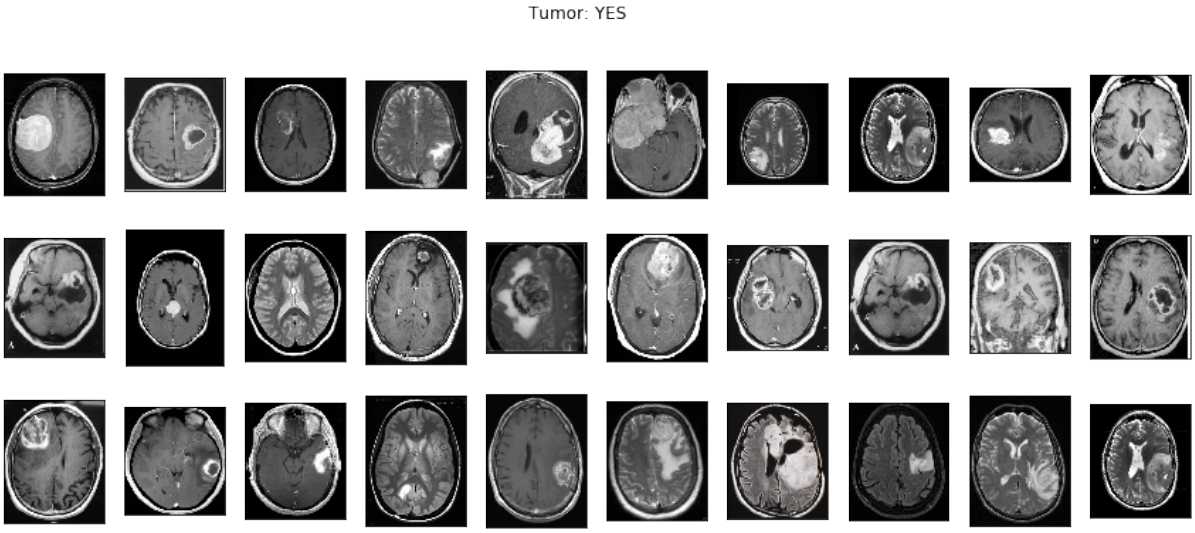
2.2 Normalizacja obrazów

Widzimy, że nasze zdjęcia mózgu są robione z różnej odległości. Spróbujemy je znormalizować tak, aby widoczny był mózg z jak najmniejszą ilością tła. W tym celu znajdziemy minimum i maksimum mózgu w pionie i poziomie.

Oto kod funkcji, która to zrobi:



Rysunek 3: Bez nowotworu w zbiorze treningowym



Rysunek 4: Z nowotworem w zbiorze treningowym

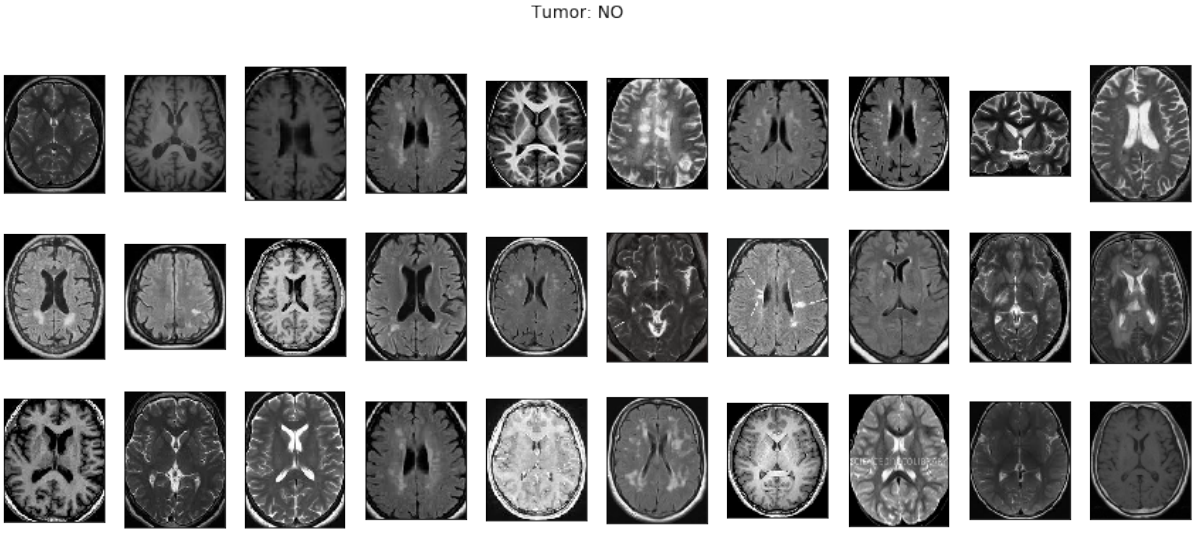
```
def crop_imgs(set_name, add_pixels_value=0):
    """
    Finds the extreme points on the image and crops the rectangular out
    of them
    """
    set_new = []
    for img in set_name:
        gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
        gray = cv2.GaussianBlur(gray, (5, 5), 0)

        # threshold the image, then perform a series of erosions +
        # dilations to remove any small regions of noise
        thresh = cv2.threshold(gray, 45, 255, cv2.THRESH_BINARY)[1]
        thresh = cv2.erode(thresh, None, iterations=2)
        thresh = cv2.dilate(thresh, None, iterations=2)

        # find contours in thresholded image, then grab the largest one
        cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL, cv2.
    CHAIN_APPROX_SIMPLE)
        cnts = imutils.grab_contours(cnts)
        c = max(cnts, key=cv2.contourArea)

        # find the extreme points
        extLeft = tuple(c[c[:, :, 0].argmin()][0])
        extRight = tuple(c[c[:, :, 0].argmax()][0])
        extTop = tuple(c[c[:, :, 1].argmin()][0])
        extBot = tuple(c[c[:, :, 1].argmax()][0])

        ADD_PIXELS = add_pixels_value
        new_img = img[extTop[1]-ADD_PIXELS:extBot[1]+ADD_PIXELS, extLeft
    [0]-ADD_PIXELS:extRight[0]+ADD_PIXELS].copy()
        set_new.append(new_img)

    return np.array(set_new)
```

Nasze obrazy po wstępnej normalizacji:
Obróbka danych jest już prawie skończona. Wystarczy jeszcze tylko ujednolicić rozmiar obrazów. Ujednoliciliśmy go do 224 na 224 pikseli.

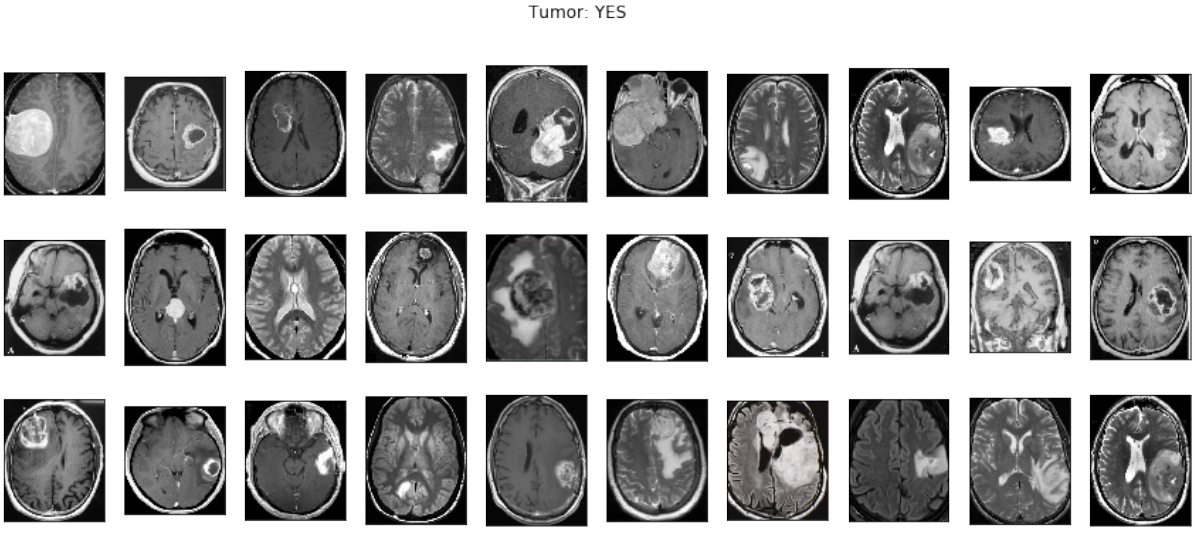
3 Tworzenie modelu

4 Wyniki

5 Wnioski



Rysunek 5: Bez nowotworu po normalizacji



Rysunek 6: Z nowotworem po normalizacji