# Inteligencja obliczeniowa - Projekt 2

Mateusz Stapaj

7 czerwca 2022

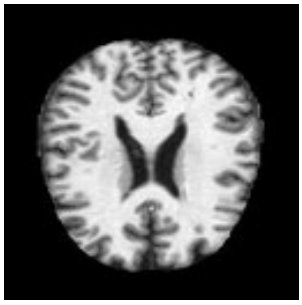
## 1 Wstęp

Projekt polega na znalezieniu algorytmu, który będzie poprawnie klasyfikował zdjęcia według kategorii. Do rozwiązania tego problemu wykorzystałem sieć konwolucyjną w różnych wersjach.

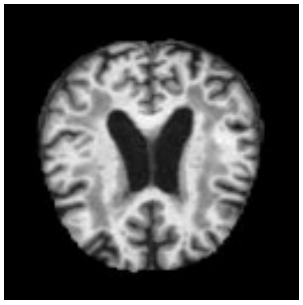
## 2 Przedstawienie problemu

Wybrałem problem sklasyfikowania stanu zaawansowania choroby Alzheimera na podstawie zdjęć zrobionych za pomocą obrazowania metodą rezonansu magnetycznego. Do wytrenowania modeli wykorzystałem bazę 6400 zdjęć podzielonych na cztery kategorie:
- mild demented - łagodna demencja
- moderate demnted - umiarkowana demencja
- non demented - bez demencji
- very mild demented - bardzo łagodna demencja

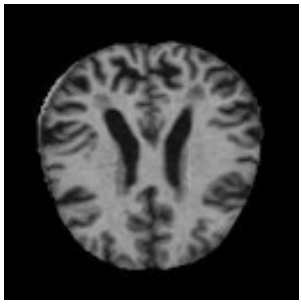
Cały zbiór podzieliłem na zbiór treningowy (80% - 5150 zdjęć) oraz zbiór testowy (20% - 1250 zdjęć). Poniżej przykładowe zdjęcia:



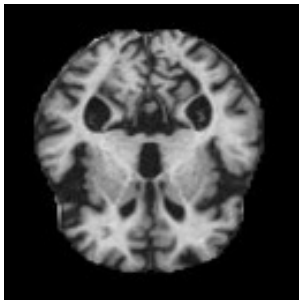
(a) Bez demencji



(b) Bardzo łagodna demencja



(c) Łagodna demencja



(d) Umiarkowana demencja

# 3 Przygotowanie danych

## 3.1 Stworzenie potrzebnych folderów

```
dirs = ["./models", "./wrong_predicted", "./divided_data"]

for i in dirs:
    dir = os.path.join(i)
    if not os.path.exists(dir):
        os.mkdir(dir)

src_directory = './data'
res = []

for file in listdir(src_directory):
    if file.split("_")[0].split(".")[0] not in res:
        res.append(file.split("_")[0].split(".")[0])

folder = './data/'
dataset_home = 'divided_data/'
subdirs = ['train/', 'test/']
for subdir in subdirs:
    labeldirs = res
    for labldir in labeldirs:
        newdir = dataset_home + subdir + labldir
        makedirs(newdir, exist_ok=True)
```

## 3.2 Podzielenie danych na zbiór treningowy i zbiór testowy

```
seed(275036)
val_ratio = 0.20

src_directory = './data'
for file in listdir(src_directory):
    src = src_directory + '/' + file
    dst_dir = '/train/'
    if random() < val_ratio:
        dst_dir = 'test/'
    dst = dataset_home + dst_dir + file.split("_")[0].split(".")[0] + '/' + file
    copyfile(src, dst)
```

# 4 Klasyfikacja za pomocą przygotowanych modeli

## 4.1 Model podstawowy

```
    model = Sequential()
    model.add(Conv2D(32, (3, 3)))
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(1, activation='softmax'))
    opt = SGD(lr=0.001, momentum=0.9)
    model.compile(optimizer=opt, loss='binary_crossentropy', metrics=['accuracy'])
```

Model ten składa się z jeden warstwy konwolucyjnej, jednej warstwy typu max pooling, jednej warstwy spłaszczającej i jednej warstwie w pełni połączonej. Użyłem optymalizatora typu SGD (optymalizator spadku gradientu).

Wyniki:

Skuteczność: 12.22%
Czas trenowania: 186.94 sekund
Czas klasyfikacji jednego obrazka: 0.12 sekundy
Czas klasyfikacji zbioru testowego: 2.89 sekund

Macierz błędów:

$$\begin{bmatrix} 184 & 0 & 0 & 0 \\ 11 & 0 & 0 & 0 \\ 616 & 0 & 0 & 0 \\ 436 & 0 & 0 & 0 \end{bmatrix}$$

## 4.2 Modele z samouczka

### 4.2.1 Model z jednym blokiem VGG

```
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same',
                 input_shape=(224, 224, 3)))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu', kernel_initializer='he_uniform'))
model.add(Dense(1, activation='sigmoid'))
opt = SGD(lr=0.001, momentum=0.9)
model.compile(optimizer=opt, loss='binary_crossentropy', metrics=['accuracy'])
```

Model ten składa się z jednej warstwy splotowej z 32 filtrami, jednej warstwy typu max pooling, jednej warstwy spłaszczającej i z dwóch warstw w pełni połączonych. W tym modelu został użyty optymalizator SGD (optymalizator spadku gradientu).

Wyniki:

Skuteczność: 14.72%
Czas trenowania: 180.54 sekund
Czas klasyfikacji jednego obrazka: 0.09 sekundy
Czas klasyfikacji zbioru testowego: 4.81 sekund
Macierz błędów:

$$\begin{bmatrix} 184 & 0 & 0 & 0 \\ 11 & 0 & 0 & 0 \\ 616 & 0 & 0 & 0 \\ 439 & 0 & 0 & 0 \end{bmatrix}$$

### 4.2.2 Model z dwoma blokami VGG

```
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same',
                 input_shape=(224, 224, 3)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu', kernel_initializer='he_uniform'))
model.add(Dense(1, activation='sigmoid'))
opt = SGD(lr=0.001, momentum=0.9)
model.compile(optimizer=opt, loss='binary_crossentropy', metrics=['accuracy'])
```

Model ten rożni się od poprzedniego tym, że na początku jest jedna warstwa splotowa z 32 filtrami, warstwa typu max pooling, warstwa splotowa z 64 filtrami, kolejna warstwa typu max pooling.

Wyniki:

Skuteczność: 15.02%
Czas trenowania: 181.06 sekund
Czas klasyfikacji jednego obrazka: 0.08 sekundy
Czas klasyfikacji zbioru testowego: 6.36 sekund

Macierz błędów:

$$\begin{bmatrix} 184 & 0 & 0 & 0 \\ 11 & 0 & 0 & 0 \\ 616 & 0 & 0 & 0 \\ 439 & 0 & 0 & 0 \end{bmatrix}$$

### 4.2.3 Model z trzema blokami VGG

```
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same',
                 input_shape=(224, 224, 3)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu', kernel_initializer='he_uniform'))
model.add(Dense(1, activation='sigmoid'))
opt = SGD(lr=0.001, momentum=0.9)
model.compile(optimizer=opt, loss='binary_crossentropy', metrics=['accuracy'])
```

Model ten rożni się od poprzedniego tym, że na początku jest jedna warstwa splotowa z 32 filtrami, warstwa typu max pooling, warstwa splotowa z 64 filtrami, warstwa typu max pooling, warstwa splotowa z 128 filtrami, warstwa typu max pooling.

Wyniki:

Skuteczność: 15.02%
Czas trenowania: 180.91 sekund
Czas klasyfikacji jednego obrazka: 0.08 sekundy
Czas klasyfikacji zbioru testowego: 7.59 sekund
Macierz błędów:

$$\begin{bmatrix} 184 & 0 & 0 & 0 \\ 11 & 0 & 0 & 0 \\ 616 & 0 & 0 & 0 \\ 439 & 0 & 0 & 0 \end{bmatrix}$$

## 4.3 Model z warstwą typu Global Average Pooling

```
model = Sequential()
model.add(GlobalAveragePooling2D())
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(4, activation="softmax"))
model.compile(optimizer="Adam", loss=tf.keras.losses.SparseCategoricalCrossentropy(), metrics=["accuracy"])
```

Model ten składa się z jednej warstwy typu Global Average Pooling, która wylicza średnią z wszystkich warstw, warstwy spłaszczającej, dwóch warstw w pełni połączonych. W tym modelu został użyty optymalizator typu Adam, który jest bardziej skuteczny i zapobiega przeuczeniu.

Wyniki:

Skuteczność: 49.28%
Czas trenowania: 193.87 sekund
Czas klasyfikacji jednego obrazka: 0.05 sekundy
Czas klasyfikacji zbioru testowego: 1.96 sekund
Macierz błędów:

$$\begin{bmatrix} 0 & 0 & 184 & 0 \\ 0 & 0 & 11 & 0 \\ 0 & 0 & 616 & 0 \\ 0 & 0 & 439 & 0 \end{bmatrix}$$

## 4.4 Model z przeskalowaniem

```
model = Sequential()
model.add(Rescaling(1. / 255, input_shape=(224, 224, 3)))
model.add(Conv2D(16, 3, padding='same', activation='relu'))
model.add(MaxPooling2D())
model.add(Conv2D(32, 3, padding='same', activation='relu'))
model.add(MaxPooling2D())
model.add(Conv2D(64, 3, padding='same', activation='relu'))
model.add(MaxPooling2D())
model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(4, activation="softmax"))
model.compile(optimizer="Adam", loss=tf.keras.losses.SparseCategoricalCrossentropy(), metrics=["accuracy"])
```

Model ten składa się z jednej warstwy, która skaluje obraz z wartości z zakresu [0,255] na zakres [0,1]. Następnie są trzy razy po dwie warstwy splotowe i max pooling. Potem jest warstwa typu Dropout, która losowo ustawia wartości na 0. Na końcu jest warstwa spłaszczająca i dwie warstwy w pełni połączone. W tym modelu został użyty optymalizator typu Adam, który jest bardziej skuteczny i zapobiega przeuczeniu.

Wyniki:

Skuteczność: 71.92%
Czas trenowania: 189.53 sekund
Czas klasyfikacji jednego obrazka: 0.08 sekundy
Czas klasyfikacji zbioru testowego: 4.02 sekund
Macierz błędów:

$$\begin{bmatrix} 11 & 0 & 84 & 89 \\ 0 & 0 & 2 & 9 \\ 43 & 0 & 313 & 260 \\ 27 & 0 & 224 & 188 \end{bmatrix}$$

## 4.5 Najlepszy model

```
model = Sequential()
model.add(Conv2D(64, (3, 3), padding="same", activation='relu'))
model.add(MaxPooling2D())
model.add(Conv2D(32, (3, 3), padding="same", activation='relu'))
model.add(MaxPooling2D())
model.add(Conv2D(32, (2, 2), padding="same", activation='relu'))
model.add(MaxPooling2D())
model.add(Flatten())
model.add(Dense(100, activation='relu'))
model.add(Dense(50, activation='relu'))
model.add(Dense(4, activation='softmax'))
model.compile(optimizer="Adam", loss=tf.keras.losses.SparseCategoricalCrossentropy(), metrics=["accuracy"])
```

Model ten składa trzech zestawów warstw splotowych i warstw typu pooling, warstwy spłaszczającej i trzech warstw w pełni połączonych. W tym modelu został użyty optymalizator typu Adam, który jest bardziej skuteczny i zapobiega przeuczeniu.

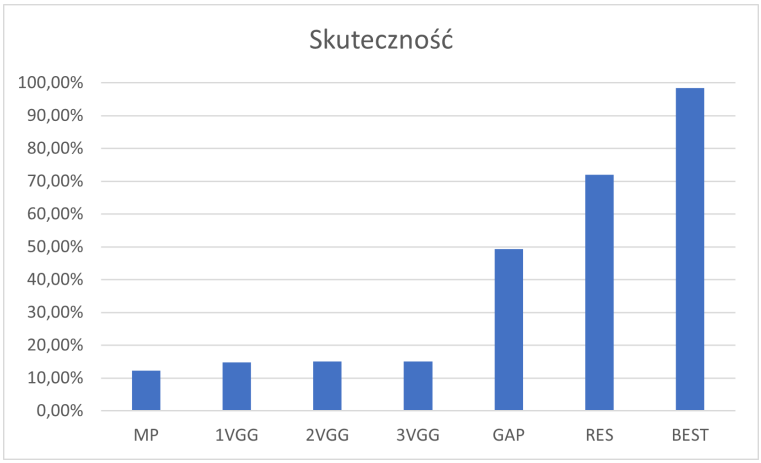
Wyniki:

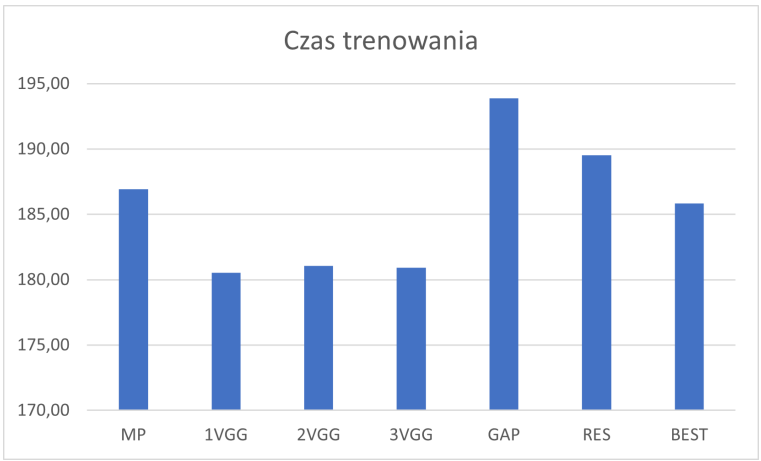
Skuteczność: 98.4%
Czas trenowania: 185.84 sekund
Czas klasyfikacji jednego obrazka: 0.08 sekundy
Czas klasyfikacji zbioru testowego: 9.08 sekund
Macierz błędów:

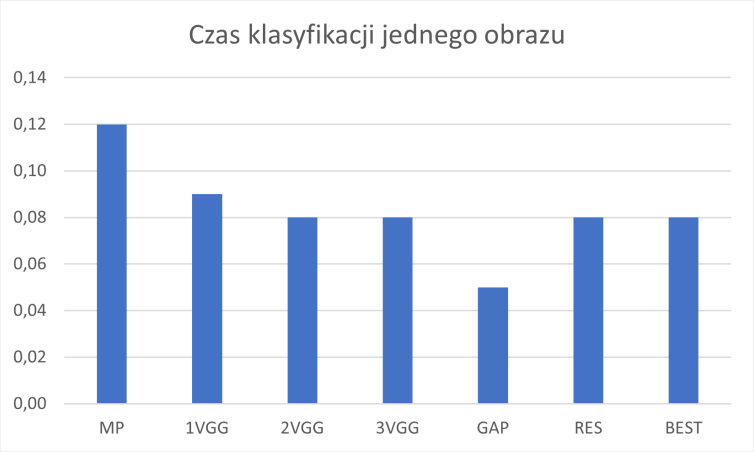
$$\begin{bmatrix} 30 & 2 & 93 & 59 \\ 05 & 0 & 5 & 1 \\ 84 & 5 & 314 & 213 \\ 68 & 3 & 207 & 161 \end{bmatrix}$$

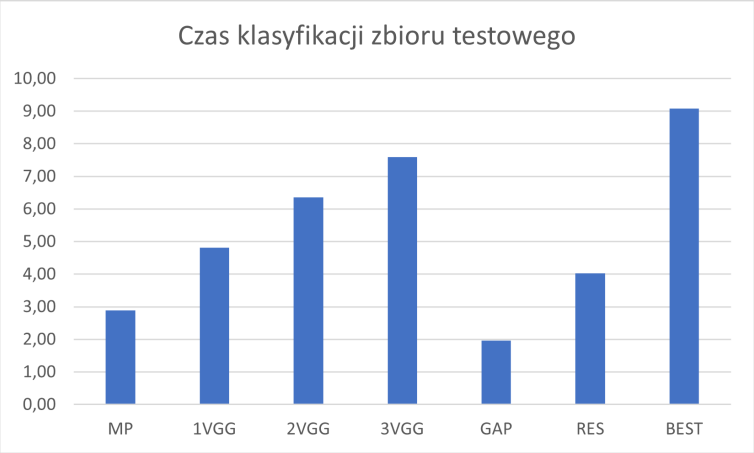
# 5 Porównanie wyników

W poniższych wykresach wykorzystałem takie skróty:
- MP - model podstawowy
- 1VGG - model z jednym blokiem VGG
- 2VGG - model z dwoma blokami VGG
- 3VGG - model z trzema blokami VGG
- GAP - model z warstwą typu Global Average Pooling
- RES - model z przeskalowaniem
- BEST - najlepszy model






# 6 Bibliografia

- https://www.kaggle.com/datasets/sachinkumar413/alzheimer-mri-dataset
- https://www.tensorflow.org/api_docs/python/tf
- https://keras.io/api/layers/
- https://keras.io/api/optimizers/
- https://keras.io/api/metrics/
- https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-to-classify-photos