# Test

November 17, 2024

### 0.1 # Test modeli

Autor: mgr inż. Grzegorz Kossakowski

## 1 1. Opis

Celem tego notebook jest przetestowanie wszystkich nauczonych architektur w celu sprawdzenia poprawności otrzymywanych przewidywań.

### 1.1 2. Wczytanie niezbędnych bibliotek

Przed rozpoczęciem pracy musimy wczytać potrzebne nam w tym notebook biblioteki.

[2]:
```
TF_ENABLE_ONEDNN_OPTS=0
import pandas as pd
from astropy.io import fits
import keras
import ipywidgets as widgets
import os
from keras.optimizers import Adam
import datetime
from sklearn.metrics import accuracy_score

from keras.applications.xception import preprocess_input as␣
 ↪preprocess_input_xception
from keras.applications.vgg16 import preprocess_input as preprocess_input_vgg
from keras.applications.resnet50 import preprocess_input as␣
 ↪preprocess_input_resnet
from keras.applications.mobilenet import preprocess_input as␣
 ↪preprocess_input_mobilenet
from keras.applications.inception_resnet_v2 import preprocess_input as␣
 ↪preprocess_input_inception_resnet
from keras.applications.inception_v3 import preprocess_input as␣
 ↪preprocess_input_inception
```

### 1.2 2. Wczytanie zbiorów testowych

Do tego etapu będziemy wykorzystywać jedynie dane testowe. Na jej podstawie ostatecznie można powiedzieć w jakim stopniu sprawdza się każda kolejna architektura

[3]:
```
hdu_test = fits.open('Data/test.fits')
x_test = hdu_test[0].data
y_test = hdu_test[1].data
x_test.shape, y_test.shape
```

[3]:
```
((3548, 256, 256, 3), (3548,))
```

### 1.3 3. Wczytanie listy architektur

W katalogu są przechowywane architektury, zostanie pobrana jej nazwa i będzie możliwe wygenerowanie dla niej test.

[4]:
```
files=os.listdir('./Models')
models = list()
for k in files:
    models.append(k.strip('.keras'))
```

### 1.4 4. Wybór architektury do testu

Każdy test odbywa się pojedynczo. Dzięki temu elementowi notebook możemy wybrać, który w danym momencie ma zostać wykonany.

[5]:
```
out = 'DNN'
rb = widgets.RadioButtons(
    options=models,
    description='Wybierz architekture do testu:',
    layout={'width':'max_content'},
    readout=True
)
display(rb)
```

```
RadioButtons(description='Wybierz architekture do testu:',␣
↪layout=Layout(width='max_content'), options=('Incep…
```

[6]:
```
out = rb.value
out
```

[6]:
```
'MobileNet'
```

[7]:
```
match out:
    case "Xception" | "Xception_full":
        x_test = preprocess_input_xception(x_test)
        print("Xception")
    case "VGG16" | "VGG16_full":
        x_test = preprocess_input_vgg(x_test)
        print("VGG16")
    case "ResNet50" | "ResNet50_full":
        x_test = preprocess_input_resnet(x_test)
        print("ResNet50")
```

```
    case "MobileNet" | "MobileNet_full":
        x_test = preprocess_input_mobilenet(x_test)
        print("MobileNet")
    case "InceptionResNet" | "InceptionResNet_full":
        x_test = preprocess_input_inception_resnet(x_test)
        print("InceptionResNet")
    case "Inception" | "Inception_full":
        x_test = preprocess_input_inception(x_test)
        print("Inception")
    case _:
        x_test = x_test / 255.0
        print("Default")
```

```
MobileNet
```

### 1.5 5. Wykonanie testu

W pierwszej kolejności jest pobierany wcześniej wybrany architektura, a następnie jest wykonywany test. Przez test rozumiemy podanie do naszych architektur danych, których model jeszcze nie widział i pobranie wyników, do jakich grup przydzielił dany obraz. Wyniki, jakie otrzymujemy oraz prawidłowe wartości, zostały zapisane w oddzielnym pliku, Dzięki temu będzie możliwość porównania różnych testów i wyciągnięcia odpowiednich wniosków.

[8]:
```
model = keras.saving.load_model('./Models/'+out+'.keras')
model_optimizer = Adam(learning_rate=0.001)
model.compile(optimizer=model_optimizer,␣
 ↪loss='sparse_categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 MobilenetV3large (Function  (None, 8, 8, 960)         2996352
 al)

 flatten (Flatten)           (None, 61440)             0

 dense (Dense)               (None, 10)                614410

=================================================================
Total params: 3610762 (13.77 MB)
Trainable params: 3411322 (13.01 MB)
Non-trainable params: 199440 (779.06 KB)
_________________________________________________________________
```

[9]:
```
predict = model.predict(x_test).argmax(axis=1)
print("Otrzymany wynik to: ",(accuracy_score(y_test, predict)*100)," %")
```

```
111/111 [==============================] - 22s 191ms/step
Otrzymany wynik to:  28.07215332581736  %
```

[10]:
```
result = pd.DataFrame()
result['predict'] = predict
result['test'] = y_test
result.to_csv('./Results/'+out+'.csv', index=False)
```

[ ]:
```

```