Result

November 17, 2024

0.1 # Analiza otrzymanych wyników

Autor: mgr inż. Grzegorz Kossakowski

[8]:
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay,␣
 ↪accuracy_score
import pathlib
import numpy as np
```

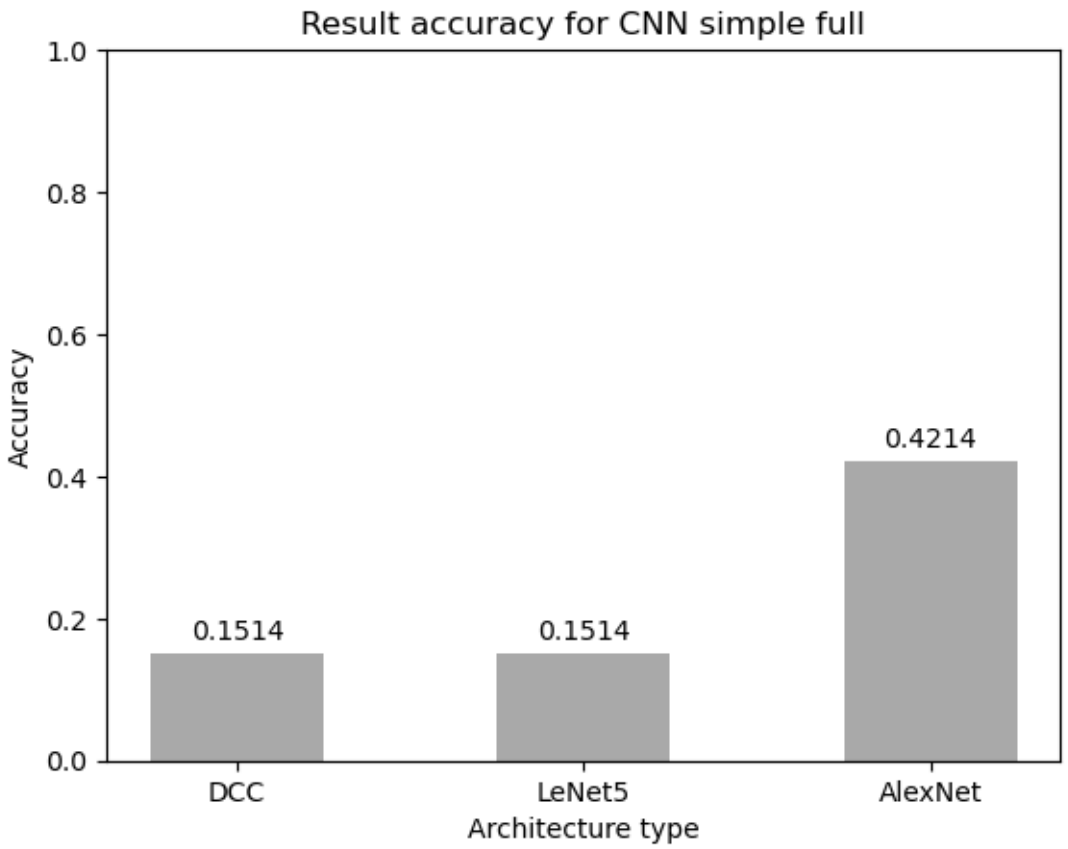
[9]:
```
features = ['Disk, Face-on, No Spiral', 'Smooth, Completely round', 'Smooth,␣
 ↪in-between round', 'Smooth, Cigar shaped', 'Disk, Edge-on, Rounded Bulge',␣
 ↪'Disk, Edge-on, Boxy Bulge','Disk, Edge-on, No Bulge','Disk, Face-on, Tight␣
 ↪Spiral', 'Disk, Face-on, Medium Spiral', 'Disk, Face-on, Loose Spiral']
```

1 1. Uczenie od początku

1.1 1.1. Proste modele

[10]:
```
resultDnnFull = pd.read_csv('./Results/DNN_full.csv')
resultLeNet5Full = pd.read_csv('./Results/LeNet5_full.csv')
resultAlexNetFull = pd.read_csv('./Results/AlexNet_full.csv')
accDNNFull = accuracy_score(resultDnnFull['test'], resultDnnFull['predict'])
accLeNet5Full = accuracy_score(resultLeNet5Full['test'],␣
 ↪resultLeNet5Full['predict'])
accAlexNetFull = accuracy_score(resultAlexNetFull['test'],␣
 ↪resultAlexNetFull['predict'])
accuracieSimpleFulls = {
    'DCC':accDNNFull,
    'LeNet5':accLeNet5Full,
    'AlexNet':accAlexNetFull
}
courses = list(accuracieSimpleFulls.keys())
values = list(accuracieSimpleFulls.values())
plt.bar(courses,values,width=0.5,label='y',color ='darkgrey', )
plt.ylim(0,1)
plt.xlabel("Architecture type")
```

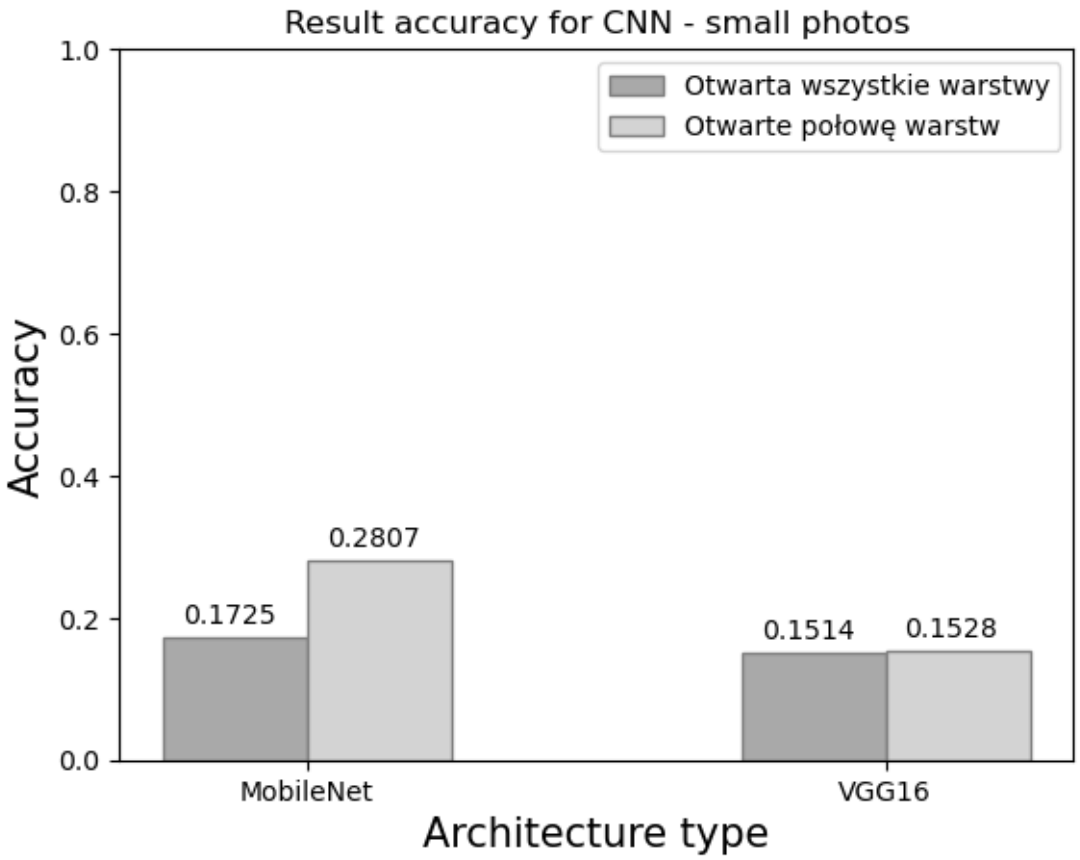
```
plt.ylabel("Accuracy")
plt.title("Result accuracy for CNN simple full")
for index, value  in enumerate(values):
    plt.text(index,(value+0.02),'{:0.4f}'.format(value),ha='center')
plt.savefig('./Picture/ResultAccuracyCNN_Article.jpeg', dpi = 900,␣
 ↪bbox_inches='tight')
plt.show()
```



2 2. Uczenie od początku

[14]:
```
resultMobileNetFull = pd.read_csv('./Results/MobileNet_full.csv')
resultMobileNet = pd.read_csv('./Results/MobileNet.csv')
resultVGG16Full = pd.read_csv('./Results/VGG16_full.csv')
resultVGG16 = pd.read_csv('./Results/VGG16.csv')
accMobileNetFull = accuracy_score(resultMobileNetFull['test'],␣
 ↪resultMobileNetFull['predict'])
accMobileNet = accuracy_score(resultMobileNet['test'],␣
 ↪resultMobileNet['predict'])
```

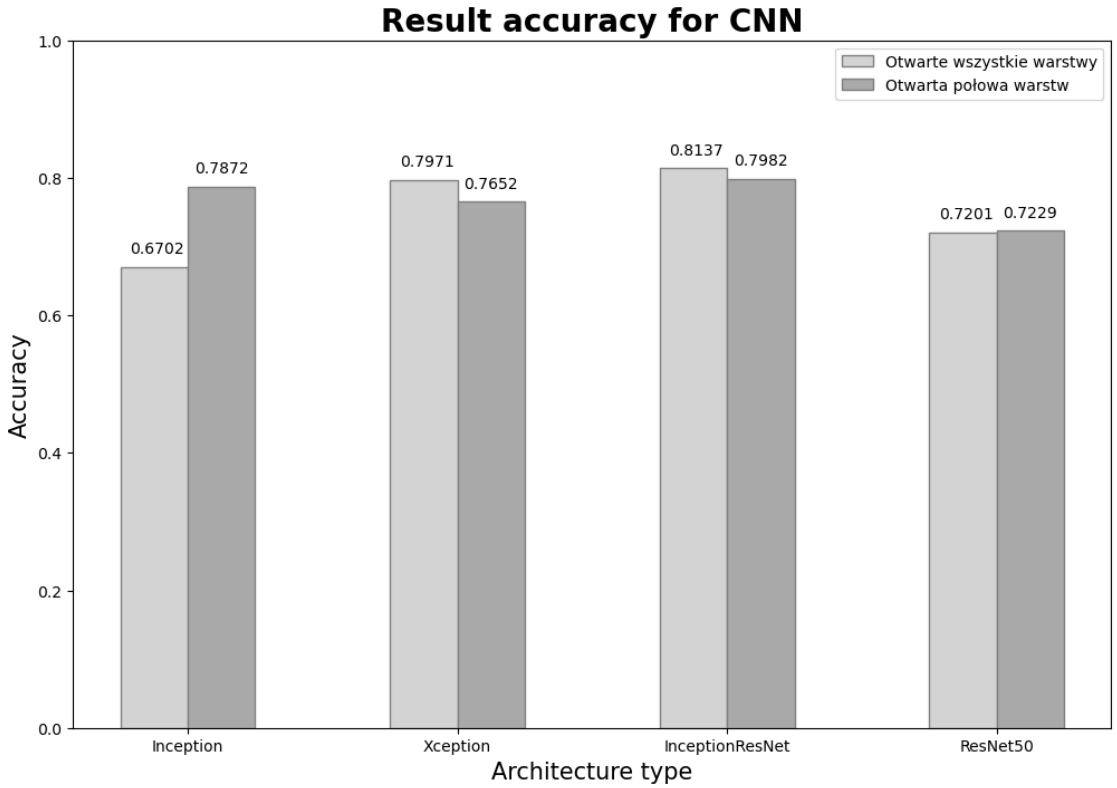
```
accVGG16Full = accuracy_score(resultVGG16Full['test'],␣
 ↪resultVGG16Full['predict'])
accVGG16 = accuracy_score(resultVGG16['test'], resultVGG16['predict'])
barWidth = 0.25
# fig = plt.subplots(figsize =(12, 8))

# set height of bar
Full = [accMobileNetFull,accVGG16Full]
Half = [accMobileNet, accVGG16]

# Set position of bar on X axis
br1 = np.arange(len(Full))
br2 = [x + barWidth for x in br1]

# Make the plot
plt.bar(br1, Full, color ='darkgrey', width = barWidth,
        edgecolor ='grey', label ='Otwarta wszystkie warstwy')
plt.bar(br2, Half, color ='lightgrey', width = barWidth,
        edgecolor ='grey', label ='Otwarte połowę warstw')

# Adding Xticks
plt.ylim(0,1)
plt.xlabel('Architecture type', fontsize = 15)
plt.ylabel('Accuracy', fontsize = 15)
plt.title("Result accuracy for CNN - small photos")
plt.xticks([r + (barWidth/2) for r in range(len(Full))],
        ['MobileNet', 'VGG16'])
for index, value  in enumerate(Full):
    plt.text(index-0.09,(value+0.02),'{:0.4f}'.format(value),ha='left')
for index, value  in enumerate(Half):
    plt.text(index+0.32,(value+0.02),'{:0.4f}'.format(value),ha='right')

plt.legend()
plt.savefig('./Picture/ResultAccuracyCNN-SmallPhotos.jpeg', dpi = 900,␣
 ↪bbox_inches='tight')
plt.show()
```



3 3. Uczone od początku

[12]:
```
resultInception = pd.read_csv('./Results/Inception.csv')
resultInceptionFull = pd.read_csv('./Results/Inception_full.csv')
resultXception = pd.read_csv('./Results/Xception.csv')
resultXceptionFull = pd.read_csv('./Results/Xception_full.csv')
resultInceptionResNet = pd.read_csv('./Results/InceptionResNet.csv')
resultInceptionResNetFull = pd.read_csv('./Results/InceptionResNet_full.csv')
resultResNet50 = pd.read_csv('./Results/ResNet50.csv')
resultResNet50Full = pd.read_csv('./Results/ResNet50_full.csv')
accInception = accuracy_score(resultInception['test'],␣
 ↪resultInception['predict'])
accInceptionFull = accuracy_score(resultInceptionFull['test'],␣
 ↪resultInceptionFull['predict'])
accXception = accuracy_score(resultXception['test'], resultXception['predict'])
accXceptionFull = accuracy_score(resultXceptionFull['test'],␣
 ↪resultXceptionFull['predict'])
```

```
accInceptionResNet = accuracy_score(resultInceptionResNet['test'],␣
 ↪resultInceptionResNet['predict'])
accInceptionResNetFull = accuracy_score(resultInceptionResNetFull['test'],␣
 ↪resultInceptionResNetFull['predict'])
accResNet50 = accuracy_score(resultResNet50['test'], resultResNet50['predict'])
accResNet50Full = accuracy_score(resultResNet50Full['test'],␣
 ↪resultResNet50Full['predict'])


barWidth = 0.25
fig = plt.subplots(figsize =(12, 8))

# set height of bar
Full = [accInceptionFull,accXceptionFull,accInceptionResNetFull,accResNet50Full]
Half = [accInception,accXception,accInceptionResNet,accResNet50]

# Set position of bar on X axis
br1 = np.arange(len(Full))
br2 = [x + barWidth for x in br1]

# Make the plot
plt.bar(br1, Full, width = barWidth, color ='lightgray',edgecolor ='grey',␣
 ↪label ='Otwarte wszystkie warstwy')
plt.bar(br2, Half, width = barWidth,color ='darkgray',edgecolor ='grey', label␣
 ↪='Otwarta połowa warstw')

# Adding Xticks
plt.ylim(0,1)
plt.xlabel('Architecture type', fontsize = 15)
plt.ylabel('Accuracy', fontsize = 15)
plt.title("Result accuracy for CNN", fontweight ='bold', fontsize = 20)
plt.xticks([r + (barWidth/2) for r in range(len(Half))],
        ['Inception', 'Xception','InceptionResNet','ResNet50'])
for index, value  in enumerate(Full):
    plt.text(index-0.09,(value+0.02),'{:0.4f}'.format(value),ha='left')
for index, value  in enumerate(Half):
    plt.text(index+0.35,(value+0.02),'{:0.4f}'.format(value),ha='right')

plt.legend()
plt.savefig('./Picture/ResultAccuracyCNN.jpeg', dpi = 900, bbox_inches='tight')
plt.show()
```



3.1 Ostateczne wyniki dla modeli, które były uczone od początku

[154]: