transfer_learning_klekowski_kruczkiewicz_makarewicz

June 25, 2023

1 Wpływ transfer learningu na sieci neuronowe.

Autorzy: Adam Klekowski, Paweł Kruczkiewicz, Piotr Makarewicz

---

Transfer learning to technika, w której model uczenia maszynowego wykorzystuje wiedzę i umiejętności nabyte na jednym zadaniu do rozwiązania innego, pokrewnego zadania. Wykonuje się ją poprzez dodanie warstwy gęsto połączonej na koniec modelu.

Fine-tuning to proces dostosowywania modelu nauczonego przy użyciu transfer learningu poprzez dotrenowanie kilku ostatnich bądź wszystkich warstw do konkretnej domeny lub problemu. Robi się to w celu uzyskania lepszej wydajności dla zbioru innego niż oryginalny.

Te techniki stosuje się, aby przyspieszyć proces uczenia, wykorzystać wiedzę z dużych zbiorów danych lub zamodelować złożone relacje, gdy nie ma wystarczająco dużo danych do treningu modelu od podstaw.

1.1 Przygotowanie

1.1.1 Import bibliotek

Do zrealizowania zadania użyliśmy biblioteki tensorflow. Obliczenia wykonywano na

[ ]:
```
import os
import time
import shutil
import random

import numpy as np
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
import seaborn as sns
```

[ ]:
```
gpus = tf.config.list_physical_devices('GPU')
print("Num GPUs Available: ", len(gpus))
if len(gpus) > 0:
    tf.config.experimental.set_memory_growth(gpus[0], True)
```

```
Num GPUs Available:  1
```

1.1.2 Funkcje pomocnicze

[ ]:
```
## consts
DATA_FOLDER="data-trimmed"

INPUT_SHAPE = (256, 256, 3)
BATCH_SIZE = 32
```

[ ]:
```
class TimeHistory(keras.callbacks.Callback):
    def on_train_begin(self, logs={}):
        self.times = []

    def on_epoch_begin(self, epoch, logs={}):
        self.epoch_time_start = time.time()

    def on_epoch_end(self, epoch, logs={}):
        self.times.append(time.time() - self.epoch_time_start)

def plot_train_results(history, times, title):
    h = history.history
    loss, acc, val_loss, val_acc = h["loss"], h["categorical_accuracy"],␣
 ↪h["val_loss"], h["val_categorical_accuracy"]
    avg_epoch_time = np.round(np.mean(times), 1)
    x = np.arange(len(loss)) + 1

    fig, (ax1, ax2) = plt.subplots(1, 2)
    fig.suptitle(f"{title} [Avg epoch time: {avg_epoch_time} s]")

    ax1.set_title("accuracy")
    ax1.plot(x, acc, label="Test")
    ax1.plot(x, val_acc, label="Validation")
    ax1.set_ylim([0, 1])
    ax1.legend()

    ax2.set_title("loss")
    ax2.plot(x, loss, label="Test")
    ax2.plot(x, val_loss, label="Validation")
    ax2.set_ylim([0, 5])
    ax2.legend()

def train_and_check_model(model, model_name, t_ds, v_ds, epochs=150,␣
 ↪lr_scale=1):
    """Trains the model and plots training results"""
    time_history = TimeHistory()

    model.compile(optimizer=keras.optimizers.Nadam(learning_rate=0.
 ↪001*lr_scale),
```

```
                  loss="categorical_crossentropy", metrics=[keras.metrics.
↪CategoricalAccuracy()])
    history = model.fit(t_ds,
                        validation_data=v_ds,
                        batch_size=BATCH_SIZE,
                        epochs=epochs,
                        callbacks=[time_history])

    model.save(f"./models/{model_name}")
    plot_train_results(history, time_history.times, model_name)
```

1.1.3 Pobranie i przygotowanie zbioru

Wybraliśmy zbiór Food-101, który zawiera zdjęcia 101 różnych rodzajów jedzenia. Z tego zbiory wybraliśmy 20 klas i z każdej klasy wzięlismy 256 próbek. Następnie zbiór podzielono na zbiór treningowy i walidacyjny w proporcji 4:1.

[ ]:
```
! wget http://data.vision.ee.ethz.ch/cvl/food-101.tar.gz
! tar xzvf food-101.tar.gz
! mv food-101 data & rm food-101.tar.gz
```

[ ]:
```
SAMPLES_PER_CLASS = 256
OLD_FOLDER = "data"
NEW_FOLDER = "data-trimmed"
CHOSEN_CLASSES = {'apple_pie', 'pizza', 'hamburger', 'spaghetti_bolognese',␣
 ↪'chocolate_cake', 'hot_dog', 'ice_cream',
                  'carrot_cake', 'chicken_curry', 'churros', 'falafel',␣
 ↪'fish_and_chips', 'french_fries', 'hummus',
                  'greek_salad', 'panna_cotta', 'nachos', 'lasagna', 'tacos',␣
 ↪'risotto'}
NUM_CLASSES = len(CHOSEN_CLASSES)

print(f'There are {NUM_CLASSES} chosen classes:')
for idx, class_name in enumerate(CHOSEN_CLASSES):
    print(f"{idx:2}. {class_name}")

if not os.path.exists(NEW_FOLDER):
    os.mkdir(NEW_FOLDER)

for class_label in CHOSEN_CLASSES:
    old_class_dir = f"{OLD_FOLDER}/{class_label}"
    new_class_dir = f"{NEW_FOLDER}/{class_label}"

    if not os.path.exists(new_class_dir):
        os.mkdir(new_class_dir)
        # print(f"Creating {new_class_dir}")
```

```
    list_of_samples = os.listdir(old_class_dir)
    trimmed_samples = random.sample(list_of_samples, SAMPLES_PER_CLASS)
    # print(f"Number of samples: {len(trimmed_samples)}")

    for sample_name in trimmed_samples:
        shutil.copyfile(f"{old_class_dir}/{sample_name}", f"{new_class_dir}/
↪{sample_name}")
        # print(f"Copying {sample_name} to {new_class_dir}")
```

```
There are 20 chosen classes:
 0. lasagna
 1. french_fries
 2. hamburger
 3. carrot_cake
 4. hot_dog
 5. panna_cotta
 6. greek_salad
 7. chicken_curry
 8. hummus
 9. apple_pie
10. fish_and_chips
11. nachos
12. ice_cream
13. spaghetti_bolognese
14. churros
15. chocolate_cake
16. pizza
17. tacos
18. falafel
19. risotto
```

[ ]:
```
train_ds, validation_ds = keras.utils.image_dataset_from_directory(
    directory=DATA_FOLDER,
    label_mode='categorical',
    image_size=(256, 256),
    validation_split=0.2,
    subset="both",
    seed=21)
```

```
Found 5120 files belonging to 20 classes.
Using 4096 files for training.
Using 1024 files for validation.
```

[ ]:
```
labels = sorted(os.listdir(DATA_FOLDER))

plt.figure(figsize=(10,10))

for (batch_of_images, batch_of_labels) in train_ds.take(1):
```

```
    batch_of_images = batch_of_images[:9]
    batch_of_labels = batch_of_labels[:9]
    for i, (image, inferred_label) in enumerate(zip(batch_of_images, batch_of_labels)):
        image = image / 255
        label = labels[np.nonzero(np.array(inferred_label))[0][0]]
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(image)
        plt.title(label)
        plt.axis("off")
```

1.2 EfficientNetV2B0

1.2.1 Użycie EfficientNetV2B0

Jako model wykorzystaliśmy model EfficientNet wytrenowany na zbiorze ImageNet.

[ ]:
```
efficient_net = keras.applications.EfficientNetV2B0(
    weights='imagenet',  # Load weights pre-trained on ImageNet.
    input_shape=(256, 256, 3),
    include_top=False,
    include_preprocessing=False)  # Do not include the ImageNet classifier at␣
 ↪the top.

efficient_net.summary()
```

```
Model: "efficientnetv2-b0"

________________________________________________________________________________
__________________
 Layer (type)                   Output Shape         Param #     Connected to
================================================================================
==================
 input_2 (InputLayer)           [(None, 256, 256, 3  0           []
                                )]

 stem_conv (Conv2D)             (None, 128, 128, 32  864
['input_2[0][0]']
                                )

 stem_bn (BatchNormalization)   (None, 128, 128, 32  128
['stem_conv[0][0]']
                                )

 stem_activation (Activation)   (None, 128, 128, 32  0
['stem_bn[0][0]']
                                )

 block1a_project_conv (Conv2D)  (None, 128, 128, 16  4608
['stem_activation[0][0]']
________________________________________________________________________________
__________________
 Layer (type)                   Output Shape         Param #     Connected to
================================================================================
==================
 input_2 (InputLayer)           [(None, 256, 256, 3  0           []
                                )]

 stem_conv (Conv2D)             (None, 128, 128, 32  864
['input_2[0][0]']
                                )
```

```
 stem_bn (BatchNormalization)   (None, 128, 128, 32  128
['stem_conv[0][0]']
                                )

 stem_activation (Activation)   (None, 128, 128, 32  0
['stem_bn[0][0]']
                                )

 block1a_project_conv (Conv2D)  (None, 128, 128, 16  4608
['stem_activation[0][0]']
                                )

 block1a_project_bn (BatchNorma  (None, 128, 128, 16  64
['block1a_project_conv[0][0]']
 lization)                      )

 block1a_project_activation (Ac  (None, 128, 128, 16  0
['block1a_project_bn[0][0]']
 tivation)                      )

 block2a_expand_conv (Conv2D)   (None, 64, 64, 64)   9216
['block1a_project_activation[0][0
                                                                ]']

 block2a_expand_bn (BatchNormal  (None, 64, 64, 64)  256
['block2a_expand_conv[0][0]']
 ization)

 block2a_expand_activation (Act  (None, 64, 64, 64)  0
['block2a_expand_bn[0][0]']
 ivation)

 block2a_project_conv (Conv2D)  (None, 64, 64, 32)   2048
['block2a_expand_activation[0][0]
                                                                ']

 block2a_project_bn (BatchNorma  (None, 64, 64, 32)  128
['block2a_project_conv[0][0]']
 lization)

 block2b_expand_conv (Conv2D)   (None, 64, 64, 128)  36864
['block2a_project_bn[0][0]']

 block2b_expand_bn (BatchNormal  (None, 64, 64, 128)  512
['block2b_expand_conv[0][0]']
 ization)
```

```
 block2b_expand_activation (Act  (None, 64, 64, 128)  0
['block2b_expand_bn[0][0]']
 ivation)

 block2b_project_conv (Conv2D)  (None, 64, 64, 32)   4096
['block2b_expand_activation[0][0]
                                                                 ']

 block2b_project_bn (BatchNorma  (None, 64, 64, 32)  128
['block2b_project_conv[0][0]']
 lization)

 block2b_drop (Dropout)         (None, 64, 64, 32)   0
['block2b_project_bn[0][0]']

 block2b_add (Add)              (None, 64, 64, 32)   0
['block2b_drop[0][0]',
'block2a_project_bn[0][0]']

 block3a_expand_conv (Conv2D)   (None, 32, 32, 128)  36864
['block2b_add[0][0]']

 block3a_expand_bn (BatchNormal  (None, 32, 32, 128)  512
['block3a_expand_conv[0][0]']
 ization)

 block3a_expand_activation (Act  (None, 32, 32, 128)  0
['block3a_expand_bn[0][0]']
 ivation)

 block3a_project_conv (Conv2D)  (None, 32, 32, 48)   6144
['block3a_expand_activation[0][0]
                                                                 ']

 block3a_project_bn (BatchNorma  (None, 32, 32, 48)  192
['block3a_project_conv[0][0]']
 lization)

 block3b_expand_conv (Conv2D)   (None, 32, 32, 192)  82944
['block3a_project_bn[0][0]']

 block3b_expand_bn (BatchNormal  (None, 32, 32, 192)  768
['block3b_expand_conv[0][0]']
 ization)

 block3b_expand_activation (Act  (None, 32, 32, 192)  0
['block3b_expand_bn[0][0]']
 ivation)
```

```
 block3b_project_conv (Conv2D)  (None, 32, 32, 48)   9216
['block3b_expand_activation[0][0]
                                                                    ']

 block3b_project_bn (BatchNorma  (None, 32, 32, 48)  192
['block3b_project_conv[0][0]']
 lization)

 block3b_drop (Dropout)         (None, 32, 32, 48)   0
['block3b_project_bn[0][0]']

 block3b_add (Add)              (None, 32, 32, 48)   0
['block3b_drop[0][0]',
'block3a_project_bn[0][0]']

 block4a_expand_conv (Conv2D)   (None, 32, 32, 192)  9216
['block3b_add[0][0]']

 block4a_expand_bn (BatchNormal  (None, 32, 32, 192)  768
['block4a_expand_conv[0][0]']
 ization)

 block4a_expand_activation (Act  (None, 32, 32, 192)  0
['block4a_expand_bn[0][0]']
 ivation)

 block4a_dwconv2 (DepthwiseConv  (None, 16, 16, 192)  1728
['block4a_expand_activation[0][0]
 2D)                                                                ']

 block4a_bn (BatchNormalization  (None, 16, 16, 192)  768
['block4a_dwconv2[0][0]']
 )

 block4a_activation (Activation  (None, 16, 16, 192)  0
['block4a_bn[0][0]']
 )

 block4a_se_squeeze (GlobalAver  (None, 192)          0
['block4a_activation[0][0]']
 agePooling2D)

 block4a_se_reshape (Reshape)   (None, 1, 1, 192)    0
['block4a_se_squeeze[0][0]']

 block4a_se_reduce (Conv2D)     (None, 1, 1, 12)     2316
['block4a_se_reshape[0][0]']
```

```
 block4a_se_expand (Conv2D)     (None, 1, 1, 192)    2496
['block4a_se_reduce[0][0]']

 block4a_se_excite (Multiply)   (None, 16, 16, 192)  0
['block4a_activation[0][0]',
'block4a_se_expand[0][0]']

 block4a_project_conv (Conv2D)  (None, 16, 16, 96)   18432
['block4a_se_excite[0][0]']

 block4a_project_bn (BatchNorma  (None, 16, 16, 96)  384
['block4a_project_conv[0][0]']
 lization)

 block4b_expand_conv (Conv2D)   (None, 16, 16, 384)  36864
['block4a_project_bn[0][0]']

 block4b_expand_bn (BatchNormal  (None, 16, 16, 384)  1536
['block4b_expand_conv[0][0]']
 ization)

 block4b_expand_activation (Act  (None, 16, 16, 384)  0
['block4b_expand_bn[0][0]']
 ivation)

 block4b_dwconv2 (DepthwiseConv  (None, 16, 16, 384)  3456
['block4b_expand_activation[0][0]
 2D)                                                           ']

 block4b_bn (BatchNormalization  (None, 16, 16, 384)  1536
['block4b_dwconv2[0][0]']
 )

 block4b_activation (Activation  (None, 16, 16, 384)  0
['block4b_bn[0][0]']
 )

 block4b_se_squeeze (GlobalAver  (None, 384)         0
['block4b_activation[0][0]']
 agePooling2D)

 block4b_se_reshape (Reshape)   (None, 1, 1, 384)    0
['block4b_se_squeeze[0][0]']

 block4b_se_reduce (Conv2D)     (None, 1, 1, 24)     9240
['block4b_se_reshape[0][0]']
```

```
 block4b_se_expand (Conv2D)     (None, 1, 1, 384)    9600
['block4b_se_reduce[0][0]']

 block4b_se_excite (Multiply)   (None, 16, 16, 384)  0
['block4b_activation[0][0]',
'block4b_se_expand[0][0]']

 block4b_project_conv (Conv2D)  (None, 16, 16, 96)   36864
['block4b_se_excite[0][0]']

 block4b_project_bn (BatchNorma  (None, 16, 16, 96)  384
['block4b_project_conv[0][0]']
 lization)

 block4b_drop (Dropout)         (None, 16, 16, 96)   0
['block4b_project_bn[0][0]']

 block4b_add (Add)              (None, 16, 16, 96)   0
['block4b_drop[0][0]',
'block4a_project_bn[0][0]']

 block4c_expand_conv (Conv2D)   (None, 16, 16, 384)  36864
['block4b_add[0][0]']

 block4c_expand_bn (BatchNormal  (None, 16, 16, 384)  1536
['block4c_expand_conv[0][0]']
 ization)

 block4c_expand_activation (Act  (None, 16, 16, 384)  0
['block4c_expand_bn[0][0]']
 ivation)

 block4c_dwconv2 (DepthwiseConv  (None, 16, 16, 384)  3456
['block4c_expand_activation[0][0]
 2D)                                                                ']

 block4c_bn (BatchNormalization  (None, 16, 16, 384)  1536
['block4c_dwconv2[0][0]']
 )

 block4c_activation (Activation  (None, 16, 16, 384)  0
['block4c_bn[0][0]']
 )

 block4c_se_squeeze (GlobalAver  (None, 384)          0
['block4c_activation[0][0]']
 agePooling2D)
```

```
 block4c_se_reshape (Reshape)   (None, 1, 1, 384)    0
['block4c_se_squeeze[0][0]']

 block4c_se_reduce (Conv2D)     (None, 1, 1, 24)     9240
['block4c_se_reshape[0][0]']

 block4c_se_expand (Conv2D)     (None, 1, 1, 384)    9600
['block4c_se_reduce[0][0]']

 block4c_se_excite (Multiply)   (None, 16, 16, 384)  0
['block4c_activation[0][0]',
'block4c_se_expand[0][0]']

 block4c_project_conv (Conv2D)  (None, 16, 16, 96)   36864
['block4c_se_excite[0][0]']

 block4c_project_bn (BatchNorma  (None, 16, 16, 96)  384
['block4c_project_conv[0][0]']
 lization)

 block4c_drop (Dropout)         (None, 16, 16, 96)   0
['block4c_project_bn[0][0]']

 block4c_add (Add)              (None, 16, 16, 96)   0
['block4c_drop[0][0]',
'block4b_add[0][0]']

 block5a_expand_conv (Conv2D)   (None, 16, 16, 576)  55296
['block4c_add[0][0]']

 block5a_expand_bn (BatchNormal  (None, 16, 16, 576)  2304
['block5a_expand_conv[0][0]']
 ization)

 block5a_expand_activation (Act  (None, 16, 16, 576)  0
['block5a_expand_bn[0][0]']
 ivation)

 block5a_dwconv2 (DepthwiseConv  (None, 16, 16, 576)  5184
['block5a_expand_activation[0][0]
 2D)                                                                    ']

 block5a_bn (BatchNormalization  (None, 16, 16, 576)  2304
['block5a_dwconv2[0][0]']
 )

 block5a_activation (Activation  (None, 16, 16, 576)  0
['block5a_bn[0][0]']
```

```
 )

 block5a_se_squeeze (GlobalAver  (None, 576)         0
['block5a_activation[0][0]']
 agePooling2D)

 block5a_se_reshape (Reshape)   (None, 1, 1, 576)    0
['block5a_se_squeeze[0][0]']

 block5a_se_reduce (Conv2D)     (None, 1, 1, 24)     13848
['block5a_se_reshape[0][0]']

 block5a_se_expand (Conv2D)     (None, 1, 1, 576)    14400
['block5a_se_reduce[0][0]']

 block5a_se_excite (Multiply)   (None, 16, 16, 576)  0
['block5a_activation[0][0]',
'block5a_se_expand[0][0]']

 block5a_project_conv (Conv2D)  (None, 16, 16, 112)  64512
['block5a_se_excite[0][0]']

 block5a_project_bn (BatchNorma  (None, 16, 16, 112)  448
['block5a_project_conv[0][0]']
 lization)

 block5b_expand_conv (Conv2D)   (None, 16, 16, 672)  75264
['block5a_project_bn[0][0]']

 block5b_expand_bn (BatchNormal  (None, 16, 16, 672)  2688
['block5b_expand_conv[0][0]']
 ization)

 block5b_expand_activation (Act  (None, 16, 16, 672)  0
['block5b_expand_bn[0][0]']
 ivation)

 block5b_dwconv2 (DepthwiseConv  (None, 16, 16, 672)  6048
['block5b_expand_activation[0][0]
 2D)                                                              ']

 block5b_bn (BatchNormalization  (None, 16, 16, 672)  2688
['block5b_dwconv2[0][0]']
 )

 block5b_activation (Activation  (None, 16, 16, 672)  0
['block5b_bn[0][0]']
 )
```

```
 block5b_se_squeeze (GlobalAver  (None, 672)         0
['block5b_activation[0][0]']
 agePooling2D)

 block5b_se_reshape (Reshape)   (None, 1, 1, 672)    0
['block5b_se_squeeze[0][0]']

 block5b_se_reduce (Conv2D)     (None, 1, 1, 28)     18844
['block5b_se_reshape[0][0]']

 block5b_se_expand (Conv2D)     (None, 1, 1, 672)    19488
['block5b_se_reduce[0][0]']

 block5b_se_excite (Multiply)   (None, 16, 16, 672)  0
['block5b_activation[0][0]',
'block5b_se_expand[0][0]']

 block5b_project_conv (Conv2D)  (None, 16, 16, 112)  75264
['block5b_se_excite[0][0]']

 block5b_project_bn (BatchNorma  (None, 16, 16, 112)  448
['block5b_project_conv[0][0]']
 lization)

 block5b_drop (Dropout)         (None, 16, 16, 112)  0
['block5b_project_bn[0][0]']

 block5b_add (Add)              (None, 16, 16, 112)  0
['block5b_drop[0][0]',
'block5a_project_bn[0][0]']

 block5c_expand_conv (Conv2D)   (None, 16, 16, 672)  75264
['block5b_add[0][0]']

 block5c_expand_bn (BatchNormal  (None, 16, 16, 672)  2688
['block5c_expand_conv[0][0]']
 ization)

 block5c_expand_activation (Act  (None, 16, 16, 672)  0
['block5c_expand_bn[0][0]']
 ivation)

 block5c_dwconv2 (DepthwiseConv  (None, 16, 16, 672)  6048
['block5c_expand_activation[0][0]
 2D)                                                              ']

 block5c_bn (BatchNormalization  (None, 16, 16, 672)  2688
```

```
['block5c_dwconv2[0][0]']
 )

 block5c_activation (Activation  (None, 16, 16, 672)  0
['block5c_bn[0][0]']
 )

 block5c_se_squeeze (GlobalAver  (None, 672)         0
['block5c_activation[0][0]']
 agePooling2D)

 block5c_se_reshape (Reshape)   (None, 1, 1, 672)    0
['block5c_se_squeeze[0][0]']

 block5c_se_reduce (Conv2D)     (None, 1, 1, 28)     18844
['block5c_se_reshape[0][0]']

 block5c_se_expand (Conv2D)     (None, 1, 1, 672)    19488
['block5c_se_reduce[0][0]']

 block5c_se_excite (Multiply)   (None, 16, 16, 672)  0
['block5c_activation[0][0]',
'block5c_se_expand[0][0]']

 block5c_project_conv (Conv2D)  (None, 16, 16, 112)  75264
['block5c_se_excite[0][0]']

 block5c_project_bn (BatchNorma  (None, 16, 16, 112)  448
['block5c_project_conv[0][0]']
 lization)

 block5c_drop (Dropout)         (None, 16, 16, 112)  0
['block5c_project_bn[0][0]']

 block5c_add (Add)              (None, 16, 16, 112)  0
['block5c_drop[0][0]',
'block5b_add[0][0]']

 block5d_expand_conv (Conv2D)   (None, 16, 16, 672)  75264
['block5c_add[0][0]']

 block5d_expand_bn (BatchNormal  (None, 16, 16, 672)  2688
['block5d_expand_conv[0][0]']
 ization)

 block5d_expand_activation (Act  (None, 16, 16, 672)  0
['block5d_expand_bn[0][0]']
 ivation)
```

```
 block5d_dwconv2 (DepthwiseConv  (None, 16, 16, 672)  6048
['block5d_expand_activation[0][0]
 2D)                                                              ']

 block5d_bn (BatchNormalization  (None, 16, 16, 672)  2688
['block5d_dwconv2[0][0]']
 )

 block5d_activation (Activation  (None, 16, 16, 672)  0
['block5d_bn[0][0]']
 )

 block5d_se_squeeze (GlobalAver  (None, 672)         0
['block5d_activation[0][0]']
 agePooling2D)

 block5d_se_reshape (Reshape)   (None, 1, 1, 672)    0
['block5d_se_squeeze[0][0]']

 block5d_se_reduce (Conv2D)     (None, 1, 1, 28)     18844
['block5d_se_reshape[0][0]']

 block5d_se_expand (Conv2D)     (None, 1, 1, 672)    19488
['block5d_se_reduce[0][0]']

 block5d_se_excite (Multiply)   (None, 16, 16, 672)  0
['block5d_activation[0][0]',
'block5d_se_expand[0][0]']

 block5d_project_conv (Conv2D)  (None, 16, 16, 112)  75264
['block5d_se_excite[0][0]']

 block5d_project_bn (BatchNorma  (None, 16, 16, 112)  448
['block5d_project_conv[0][0]']
 lization)

 block5d_drop (Dropout)         (None, 16, 16, 112)  0
['block5d_project_bn[0][0]']

 block5d_add (Add)              (None, 16, 16, 112)  0
['block5d_drop[0][0]',
'block5c_add[0][0]']

 block5e_expand_conv (Conv2D)   (None, 16, 16, 672)  75264
['block5d_add[0][0]']

 block5e_expand_bn (BatchNormal  (None, 16, 16, 672)  2688
```

```
['block5e_expand_conv[0][0]']
 ization)

 block5e_expand_activation (Act  (None, 16, 16, 672)  0
['block5e_expand_bn[0][0]']
 ivation)

 block5e_dwconv2 (DepthwiseConv  (None, 16, 16, 672)  6048
['block5e_expand_activation[0][0]
 2D)                                                              ']

 block5e_bn (BatchNormalization  (None, 16, 16, 672)  2688
['block5e_dwconv2[0][0]']
 )

 block5e_activation (Activation  (None, 16, 16, 672)  0
['block5e_bn[0][0]']
 )

 block5e_se_squeeze (GlobalAver  (None, 672)         0
['block5e_activation[0][0]']
 agePooling2D)

 block5e_se_reshape (Reshape)   (None, 1, 1, 672)    0
['block5e_se_squeeze[0][0]']

 block5e_se_reduce (Conv2D)     (None, 1, 1, 28)     18844
['block5e_se_reshape[0][0]']

 block5e_se_expand (Conv2D)     (None, 1, 1, 672)    19488
['block5e_se_reduce[0][0]']

 block5e_se_excite (Multiply)   (None, 16, 16, 672)  0
['block5e_activation[0][0]',
'block5e_se_expand[0][0]']

 block5e_project_conv (Conv2D)  (None, 16, 16, 112)  75264
['block5e_se_excite[0][0]']

 block5e_project_bn (BatchNorma  (None, 16, 16, 112)  448
['block5e_project_conv[0][0]']
 lization)

 block5e_drop (Dropout)         (None, 16, 16, 112)  0
['block5e_project_bn[0][0]']

 block5e_add (Add)              (None, 16, 16, 112)  0
['block5e_drop[0][0]',
```

```
'block5d_add[0][0]']

 block6a_expand_conv (Conv2D)   (None, 16, 16, 672)  75264
['block5e_add[0][0]']

 block6a_expand_bn (BatchNormal  (None, 16, 16, 672)  2688
['block6a_expand_conv[0][0]']
 ization)

 block6a_expand_activation (Act  (None, 16, 16, 672)  0
['block6a_expand_bn[0][0]']
 ivation)

 block6a_dwconv2 (DepthwiseConv  (None, 8, 8, 672)   6048
['block6a_expand_activation[0][0]
 2D)                                                            ']

 block6a_bn (BatchNormalization  (None, 8, 8, 672)   2688
['block6a_dwconv2[0][0]']
 )

 block6a_activation (Activation  (None, 8, 8, 672)   0
['block6a_bn[0][0]']
 )

 block6a_se_squeeze (GlobalAver  (None, 672)         0
['block6a_activation[0][0]']
 agePooling2D)

 block6a_se_reshape (Reshape)   (None, 1, 1, 672)    0
['block6a_se_squeeze[0][0]']

 block6a_se_reduce (Conv2D)     (None, 1, 1, 28)     18844
['block6a_se_reshape[0][0]']

 block6a_se_expand (Conv2D)     (None, 1, 1, 672)    19488
['block6a_se_reduce[0][0]']

 block6a_se_excite (Multiply)   (None, 8, 8, 672)    0
['block6a_activation[0][0]',
'block6a_se_expand[0][0]']

 block6a_project_conv (Conv2D)  (None, 8, 8, 192)    129024
['block6a_se_excite[0][0]']

 block6a_project_bn (BatchNorma  (None, 8, 8, 192)   768
['block6a_project_conv[0][0]']
 lization)
```

```
 block6b_expand_conv (Conv2D)   (None, 8, 8, 1152)   221184
['block6a_project_bn[0][0]']

 block6b_expand_bn (BatchNormal  (None, 8, 8, 1152)  4608
['block6b_expand_conv[0][0]']
 ization)

 block6b_expand_activation (Act  (None, 8, 8, 1152)  0
['block6b_expand_bn[0][0]']
 ivation)

 block6b_dwconv2 (DepthwiseConv  (None, 8, 8, 1152)  10368
['block6b_expand_activation[0][0]
 2D)                                                             ']

 block6b_bn (BatchNormalization  (None, 8, 8, 1152)  4608
['block6b_dwconv2[0][0]']
 )

 block6b_activation (Activation  (None, 8, 8, 1152)  0
['block6b_bn[0][0]']
 )

 block6b_se_squeeze (GlobalAver  (None, 1152)        0
['block6b_activation[0][0]']
 agePooling2D)

 block6b_se_reshape (Reshape)   (None, 1, 1, 1152)   0
['block6b_se_squeeze[0][0]']

 block6b_se_reduce (Conv2D)     (None, 1, 1, 48)     55344
['block6b_se_reshape[0][0]']

 block6b_se_expand (Conv2D)     (None, 1, 1, 1152)   56448
['block6b_se_reduce[0][0]']

 block6b_se_excite (Multiply)   (None, 8, 8, 1152)   0
['block6b_activation[0][0]',
'block6b_se_expand[0][0]']

 block6b_project_conv (Conv2D)  (None, 8, 8, 192)    221184
['block6b_se_excite[0][0]']

 block6b_project_bn (BatchNorma  (None, 8, 8, 192)   768
['block6b_project_conv[0][0]']
 lization)
```

```
 block6b_drop (Dropout)         (None, 8, 8, 192)    0
['block6b_project_bn[0][0]']

 block6b_add (Add)              (None, 8, 8, 192)    0
['block6b_drop[0][0]',
'block6a_project_bn[0][0]']

 block6c_expand_conv (Conv2D)   (None, 8, 8, 1152)   221184
['block6b_add[0][0]']

 block6c_expand_bn (BatchNormal  (None, 8, 8, 1152)  4608
['block6c_expand_conv[0][0]']
 ization)

 block6c_expand_activation (Act  (None, 8, 8, 1152)  0
['block6c_expand_bn[0][0]']
 ivation)

 block6c_dwconv2 (DepthwiseConv  (None, 8, 8, 1152)  10368
['block6c_expand_activation[0][0]
 2D)                                                             ']

 block6c_bn (BatchNormalization  (None, 8, 8, 1152)  4608
['block6c_dwconv2[0][0]']
 )

 block6c_activation (Activation  (None, 8, 8, 1152)  0
['block6c_bn[0][0]']
 )

 block6c_se_squeeze (GlobalAver  (None, 1152)        0
['block6c_activation[0][0]']
 agePooling2D)

 block6c_se_reshape (Reshape)   (None, 1, 1, 1152)   0
['block6c_se_squeeze[0][0]']

 block6c_se_reduce (Conv2D)     (None, 1, 1, 48)     55344
['block6c_se_reshape[0][0]']

 block6c_se_expand (Conv2D)     (None, 1, 1, 1152)   56448
['block6c_se_reduce[0][0]']

 block6c_se_excite (Multiply)   (None, 8, 8, 1152)   0
['block6c_activation[0][0]',
'block6c_se_expand[0][0]']

 block6c_project_conv (Conv2D)  (None, 8, 8, 192)    221184
```

```
['block6c_se_excite[0][0]']

 block6c_project_bn (BatchNorma  (None, 8, 8, 192)   768
['block6c_project_conv[0][0]']
 lization)

 block6c_drop (Dropout)         (None, 8, 8, 192)    0
['block6c_project_bn[0][0]']

 block6c_add (Add)              (None, 8, 8, 192)    0
['block6c_drop[0][0]',
'block6b_add[0][0]']

 block6d_expand_conv (Conv2D)   (None, 8, 8, 1152)   221184
['block6c_add[0][0]']

 block6d_expand_bn (BatchNormal  (None, 8, 8, 1152)  4608
['block6d_expand_conv[0][0]']
 ization)

 block6d_expand_activation (Act  (None, 8, 8, 1152)  0
['block6d_expand_bn[0][0]']
 ivation)

 block6d_dwconv2 (DepthwiseConv  (None, 8, 8, 1152)  10368
['block6d_expand_activation[0][0]
 2D)                                                                ']

 block6d_bn (BatchNormalization  (None, 8, 8, 1152)  4608
['block6d_dwconv2[0][0]']
 )

 block6d_activation (Activation  (None, 8, 8, 1152)  0
['block6d_bn[0][0]']
 )

 block6d_se_squeeze (GlobalAver  (None, 1152)        0
['block6d_activation[0][0]']
 agePooling2D)

 block6d_se_reshape (Reshape)   (None, 1, 1, 1152)   0
['block6d_se_squeeze[0][0]']

 block6d_se_reduce (Conv2D)     (None, 1, 1, 48)     55344
['block6d_se_reshape[0][0]']

 block6d_se_expand (Conv2D)     (None, 1, 1, 1152)   56448
['block6d_se_reduce[0][0]']
```

```
 block6d_se_excite (Multiply)   (None, 8, 8, 1152)   0
['block6d_activation[0][0]',
'block6d_se_expand[0][0]']

 block6d_project_conv (Conv2D)  (None, 8, 8, 192)    221184
['block6d_se_excite[0][0]']

 block6d_project_bn (BatchNorma  (None, 8, 8, 192)   768
['block6d_project_conv[0][0]']
 lization)

 block6d_drop (Dropout)         (None, 8, 8, 192)    0
['block6d_project_bn[0][0]']

 block6d_add (Add)              (None, 8, 8, 192)    0
['block6d_drop[0][0]',
'block6c_add[0][0]']

 block6e_expand_conv (Conv2D)   (None, 8, 8, 1152)   221184
['block6d_add[0][0]']

 block6e_expand_bn (BatchNormal  (None, 8, 8, 1152)  4608
['block6e_expand_conv[0][0]']
 ization)

 block6e_expand_activation (Act  (None, 8, 8, 1152)  0
['block6e_expand_bn[0][0]']
 ivation)

 block6e_dwconv2 (DepthwiseConv  (None, 8, 8, 1152)  10368
['block6e_expand_activation[0][0]
 2D)                                                                '] 

 block6e_bn (BatchNormalization  (None, 8, 8, 1152)  4608
['block6e_dwconv2[0][0]']
 )

 block6e_activation (Activation  (None, 8, 8, 1152)  0
['block6e_bn[0][0]']
 )

 block6e_se_squeeze (GlobalAver  (None, 1152)        0
['block6e_activation[0][0]']
 agePooling2D)

 block6e_se_reshape (Reshape)   (None, 1, 1, 1152)   0
['block6e_se_squeeze[0][0]']
```

```
 block6e_se_reduce (Conv2D)     (None, 1, 1, 48)     55344
['block6e_se_reshape[0][0]']

 block6e_se_expand (Conv2D)     (None, 1, 1, 1152)   56448
['block6e_se_reduce[0][0]']

 block6e_se_excite (Multiply)   (None, 8, 8, 1152)   0
['block6e_activation[0][0]',
'block6e_se_expand[0][0]']

 block6e_project_conv (Conv2D)  (None, 8, 8, 192)    221184
['block6e_se_excite[0][0]']

 block6e_project_bn (BatchNorma  (None, 8, 8, 192)   768
['block6e_project_conv[0][0]']
 lization)

 block6e_drop (Dropout)         (None, 8, 8, 192)    0
['block6e_project_bn[0][0]']

 block6e_add (Add)              (None, 8, 8, 192)    0
['block6e_drop[0][0]',
'block6d_add[0][0]']

 block6f_expand_conv (Conv2D)   (None, 8, 8, 1152)   221184
['block6e_add[0][0]']

 block6f_expand_bn (BatchNormal  (None, 8, 8, 1152)  4608
['block6f_expand_conv[0][0]']
 ization)

 block6f_expand_activation (Act  (None, 8, 8, 1152)  0
['block6f_expand_bn[0][0]']
 ivation)

 block6f_dwconv2 (DepthwiseConv  (None, 8, 8, 1152)  10368
['block6f_expand_activation[0][0]
 2D)                                                              ']

 block6f_bn (BatchNormalization  (None, 8, 8, 1152)  4608
['block6f_dwconv2[0][0]']
 )

 block6f_activation (Activation  (None, 8, 8, 1152)  0
['block6f_bn[0][0]']
 )
```

```
 block6f_se_squeeze (GlobalAver  (None, 1152)        0
['block6f_activation[0][0]']
 agePooling2D)

 block6f_se_reshape (Reshape)   (None, 1, 1, 1152)   0
['block6f_se_squeeze[0][0]']

 block6f_se_reduce (Conv2D)     (None, 1, 1, 48)     55344
['block6f_se_reshape[0][0]']

 block6f_se_expand (Conv2D)     (None, 1, 1, 1152)   56448
['block6f_se_reduce[0][0]']

 block6f_se_excite (Multiply)   (None, 8, 8, 1152)   0
['block6f_activation[0][0]',
'block6f_se_expand[0][0]']

 block6f_project_conv (Conv2D)  (None, 8, 8, 192)    221184
['block6f_se_excite[0][0]']

 block6f_project_bn (BatchNorma  (None, 8, 8, 192)   768
['block6f_project_conv[0][0]']
 lization)

 block6f_drop (Dropout)         (None, 8, 8, 192)    0
['block6f_project_bn[0][0]']

 block6f_add (Add)              (None, 8, 8, 192)    0
['block6f_drop[0][0]',
'block6e_add[0][0]']

 block6g_expand_conv (Conv2D)   (None, 8, 8, 1152)   221184
['block6f_add[0][0]']

 block6g_expand_bn (BatchNormal  (None, 8, 8, 1152)  4608
['block6g_expand_conv[0][0]']
 ization)

 block6g_expand_activation (Act  (None, 8, 8, 1152)  0
['block6g_expand_bn[0][0]']
 ivation)

 block6g_dwconv2 (DepthwiseConv  (None, 8, 8, 1152)  10368
['block6g_expand_activation[0][0]
 2D)                                                            ']

 block6g_bn (BatchNormalization  (None, 8, 8, 1152)  4608
['block6g_dwconv2[0][0]']
```

```
 )

 block6g_activation (Activation  (None, 8, 8, 1152)  0
['block6g_bn[0][0]']
 )

 block6g_se_squeeze (GlobalAver  (None, 1152)        0
['block6g_activation[0][0]']
 agePooling2D)

 block6g_se_reshape (Reshape)   (None, 1, 1, 1152)   0
['block6g_se_squeeze[0][0]']

 block6g_se_reduce (Conv2D)     (None, 1, 1, 48)     55344
['block6g_se_reshape[0][0]']

 block6g_se_expand (Conv2D)     (None, 1, 1, 1152)   56448
['block6g_se_reduce[0][0]']

 block6g_se_excite (Multiply)   (None, 8, 8, 1152)   0
['block6g_activation[0][0]',
'block6g_se_expand[0][0]']

 block6g_project_conv (Conv2D)  (None, 8, 8, 192)    221184
['block6g_se_excite[0][0]']

 block6g_project_bn (BatchNorma  (None, 8, 8, 192)   768
['block6g_project_conv[0][0]']
 lization)

 block6g_drop (Dropout)         (None, 8, 8, 192)    0
['block6g_project_bn[0][0]']

 block6g_add (Add)              (None, 8, 8, 192)    0
['block6g_drop[0][0]',
'block6f_add[0][0]']

 block6h_expand_conv (Conv2D)   (None, 8, 8, 1152)   221184
['block6g_add[0][0]']

 block6h_expand_bn (BatchNormal  (None, 8, 8, 1152)  4608
['block6h_expand_conv[0][0]']
 ization)

 block6h_expand_activation (Act  (None, 8, 8, 1152)  0
['block6h_expand_bn[0][0]']
 ivation)
```

```
 block6h_dwconv2 (DepthwiseConv  (None, 8, 8, 1152)  10368
['block6h_expand_activation[0][0]
 2D)                                                            ']

 block6h_bn (BatchNormalization  (None, 8, 8, 1152)  4608
['block6h_dwconv2[0][0]']
 )

 block6h_activation (Activation  (None, 8, 8, 1152)  0
['block6h_bn[0][0]']
 )

 block6h_se_squeeze (GlobalAver  (None, 1152)        0
['block6h_activation[0][0]']
 agePooling2D)

 block6h_se_reshape (Reshape)   (None, 1, 1, 1152)   0
['block6h_se_squeeze[0][0]']

 block6h_se_reduce (Conv2D)     (None, 1, 1, 48)     55344
['block6h_se_reshape[0][0]']

 block6h_se_expand (Conv2D)     (None, 1, 1, 1152)   56448
['block6h_se_reduce[0][0]']

 block6h_se_excite (Multiply)   (None, 8, 8, 1152)   0
['block6h_activation[0][0]',
'block6h_se_expand[0][0]']

 block6h_project_conv (Conv2D)  (None, 8, 8, 192)    221184
['block6h_se_excite[0][0]']

 block6h_project_bn (BatchNorma  (None, 8, 8, 192)   768
['block6h_project_conv[0][0]']
 lization)

 block6h_drop (Dropout)         (None, 8, 8, 192)    0
['block6h_project_bn[0][0]']

 block6h_add (Add)              (None, 8, 8, 192)    0
['block6h_drop[0][0]',
'block6g_add[0][0]']

 top_conv (Conv2D)              (None, 8, 8, 1280)   245760
['block6h_add[0][0]']

 top_bn (BatchNormalization)    (None, 8, 8, 1280)   5120
['top_conv[0][0]']
```

```
 top_activation (Activation)    (None, 8, 8, 1280)   0
['top_bn[0][0]']

================================================================================
==================
Total params: 5,919,312
Trainable params: 5,858,704
Non-trainable params: 60,608
--------------------------------------------------------------------------------
------------------
```

[ ]:
```
DROPOUT_RATE = 0.6
L1_PENALTY = 1e-5
L2_PENALTY = 1e-5

def create_transferred_model(base_model):
    base_model.trainable = False  # Freezing the base model

    inputs = keras.Input(shape=(256, 256, 3),
                         batch_size=BATCH_SIZE)
    scaled = keras.layers.Rescaling(scale=1./255.)(inputs)

    x_base = base_model(scaled, training=False)
    gap_layer = keras.layers.GlobalAveragePooling2D()(x_base)

    dropout_layer = keras.layers.Dropout(DROPOUT_RATE)(gap_layer)

    outputs = keras.layers.Dense(NUM_CLASSES,
                                 activation="softmax",
                                 kernel_regularizer=keras.regularizers.
 ↪L1L2(l1=L1_PENALTY, l2=L2_PENALTY),
                                 )(dropout_layer)
    return keras.models.Model(inputs, outputs)
```

[ ]:
```
transferred_model = create_transferred_model(efficient_net)
transferred_model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(32, 256, 256, 3)]       0

 rescaling (Rescaling)       (32, 256, 256, 3)         0

 efficientnetv2-b0 (Function  (None, 8, 8, 1280)       5919312
 al)
```

```
 global_average_pooling2d (G  (32, 1280)              0
 lobalAveragePooling2D)

 dropout (Dropout)           (32, 1280)              0

 dense (Dense)               (32, 20)                25620

=================================================================
Total params: 5,944,932
Trainable params: 25,620
Non-trainable params: 5,919,312
_________________________________________________________________
```

1.3 Bez transfer learningu

Model bez transfer learningu wykazuje precyzję na poziomie 3%. Jest to równoważne losowaniu, czyli model bez zastosowania transfer learningu jest nieefektywny.

[ ]:
```
without_training = transferred_model
without_training.compile(optimizer="adam", loss="categorical_crossentropy",␣
 ↪metrics=[keras.metrics.CategoricalAccuracy()])
without_training.evaluate(validation_ds)
```

```
32/32 [==============================] - 12s 77ms/step - loss: 3.1183 -
categorical_accuracy: 0.0352
```

[ ]:
```
[3.118260383605957, 0.03515625]
```

1.4 Transfer learning

W pierwszych próbach transfer learningu model overfitował. Zastosowanie warstwy `dropout'u` oraz regularyzacji L2 pozwoliło zniwelować ten problem i uzyskać precyzję na poziomie 72% w porównaniu z wcześniejszą rzędu ~50%.

Przed użyciem warstwy dropoutu

Po użyciu warstwy dropoutu

[ ]:
```
train_and_check_model(transferred_model, "Transfer learning model",␣
 ↪t_ds=train_ds, v_ds=validation_ds, epochs=100, lr_scale=0.2)
```

```
Epoch 1/100
128/128 [==============================] - 23s 111ms/step - loss: 2.9732 -
categorical_accuracy: 0.1064 - val_loss: 2.5579 - val_categorical_accuracy:
0.3574
Epoch 2/100
128/128 [==============================] - 14s 102ms/step - loss: 2.4726 -
```

```
categorical_accuracy: 0.2827 - val_loss: 2.1727 - val_categorical_accuracy:
0.5088
Epoch 3/100
128/128 [==============================] - 13s 101ms/step - loss: 2.1316 -
categorical_accuracy: 0.4016 - val_loss: 1.9176 - val_categorical_accuracy:
0.5674
Epoch 4/100
128/128 [==============================] - 13s 100ms/step - loss: 1.9076 -
categorical_accuracy: 0.4697 - val_loss: 1.7433 - val_categorical_accuracy:
0.5879
Epoch 5/100
128/128 [==============================] - 14s 103ms/step - loss: 1.7407 -
categorical_accuracy: 0.5168 - val_loss: 1.6189 - val_categorical_accuracy:
0.6094
Epoch 6/100
128/128 [==============================] - 13s 99ms/step - loss: 1.6277 -
categorical_accuracy: 0.5547 - val_loss: 1.5245 - val_categorical_accuracy:
0.6279
Epoch 7/100
128/128 [==============================] - 13s 99ms/step - loss: 1.5363 -
categorical_accuracy: 0.5642 - val_loss: 1.4520 - val_categorical_accuracy:
0.6396
Epoch 8/100
128/128 [==============================] - 13s 99ms/step - loss: 1.4542 -
categorical_accuracy: 0.5938 - val_loss: 1.3964 - val_categorical_accuracy:
0.6475
Epoch 9/100
128/128 [==============================] - 13s 99ms/step - loss: 1.4046 -
categorical_accuracy: 0.6050 - val_loss: 1.3521 - val_categorical_accuracy:
0.6523
Epoch 10/100
128/128 [==============================] - 13s 100ms/step - loss: 1.3574 -
categorical_accuracy: 0.6155 - val_loss: 1.3122 - val_categorical_accuracy:
0.6572
Epoch 11/100
128/128 [==============================] - 13s 100ms/step - loss: 1.2957 -
categorical_accuracy: 0.6387 - val_loss: 1.2806 - val_categorical_accuracy:
0.6572
Epoch 12/100
128/128 [==============================] - 14s 103ms/step - loss: 1.2672 -
categorical_accuracy: 0.6350 - val_loss: 1.2525 - val_categorical_accuracy:
0.6592
Epoch 13/100
128/128 [==============================] - 14s 103ms/step - loss: 1.2381 -
categorical_accuracy: 0.6479 - val_loss: 1.2303 - val_categorical_accuracy:
0.6602
Epoch 14/100
128/128 [==============================] - 13s 101ms/step - loss: 1.2088 -
```

```
categorical_accuracy: 0.6575 - val_loss: 1.2091 - val_categorical_accuracy:
0.6670
Epoch 15/100
128/128 [==============================] - 13s 101ms/step - loss: 1.1768 -
categorical_accuracy: 0.6707 - val_loss: 1.1899 - val_categorical_accuracy:
0.6680
Epoch 16/100
128/128 [==============================] - 13s 100ms/step - loss: 1.1601 -
categorical_accuracy: 0.6699 - val_loss: 1.1723 - val_categorical_accuracy:
0.6709
Epoch 17/100
128/128 [==============================] - 14s 102ms/step - loss: 1.1386 -
categorical_accuracy: 0.6738 - val_loss: 1.1587 - val_categorical_accuracy:
0.6738
Epoch 18/100
128/128 [==============================] - 14s 101ms/step - loss: 1.1116 -
categorical_accuracy: 0.6841 - val_loss: 1.1446 - val_categorical_accuracy:
0.6836
Epoch 19/100
128/128 [==============================] - 13s 100ms/step - loss: 1.0906 -
categorical_accuracy: 0.6882 - val_loss: 1.1323 - val_categorical_accuracy:
0.6826
Epoch 20/100
128/128 [==============================] - 13s 98ms/step - loss: 1.0675 -
categorical_accuracy: 0.6995 - val_loss: 1.1221 - val_categorical_accuracy:
0.6865
Epoch 21/100
128/128 [==============================] - 13s 100ms/step - loss: 1.0584 -
categorical_accuracy: 0.6970 - val_loss: 1.1106 - val_categorical_accuracy:
0.6855
Epoch 22/100
128/128 [==============================] - 13s 100ms/step - loss: 1.0521 -
categorical_accuracy: 0.6960 - val_loss: 1.1022 - val_categorical_accuracy:
0.6875
Epoch 23/100
128/128 [==============================] - 13s 100ms/step - loss: 1.0329 -
categorical_accuracy: 0.6968 - val_loss: 1.0932 - val_categorical_accuracy:
0.6924
Epoch 24/100
128/128 [==============================] - 13s 99ms/step - loss: 1.0251 -
categorical_accuracy: 0.7092 - val_loss: 1.0857 - val_categorical_accuracy:
0.6914
Epoch 25/100
128/128 [==============================] - 13s 101ms/step - loss: 0.9958 -
categorical_accuracy: 0.7156 - val_loss: 1.0787 - val_categorical_accuracy:
0.6934
Epoch 26/100
128/128 [==============================] - 13s 100ms/step - loss: 0.9997 -
```

```
categorical_accuracy: 0.7144 - val_loss: 1.0705 - val_categorical_accuracy:
0.6963
Epoch 27/100
128/128 [==============================] - 13s 99ms/step - loss: 0.9823 -
categorical_accuracy: 0.7163 - val_loss: 1.0646 - val_categorical_accuracy:
0.6992
Epoch 28/100
128/128 [==============================] - 13s 101ms/step - loss: 0.9645 -
categorical_accuracy: 0.7244 - val_loss: 1.0586 - val_categorical_accuracy:
0.7002
Epoch 29/100
128/128 [==============================] - 14s 102ms/step - loss: 0.9577 -
categorical_accuracy: 0.7207 - val_loss: 1.0540 - val_categorical_accuracy:
0.7021
Epoch 30/100
128/128 [==============================] - 13s 100ms/step - loss: 0.9486 -
categorical_accuracy: 0.7251 - val_loss: 1.0476 - val_categorical_accuracy:
0.7051
Epoch 31/100
128/128 [==============================] - 13s 100ms/step - loss: 0.9496 -
categorical_accuracy: 0.7244 - val_loss: 1.0432 - val_categorical_accuracy:
0.7021
Epoch 32/100
128/128 [==============================] - 14s 102ms/step - loss: 0.9291 -
categorical_accuracy: 0.7305 - val_loss: 1.0379 - val_categorical_accuracy:
0.7041
Epoch 33/100
128/128 [==============================] - 13s 100ms/step - loss: 0.9251 -
categorical_accuracy: 0.7358 - val_loss: 1.0328 - val_categorical_accuracy:
0.7061
Epoch 34/100
128/128 [==============================] - 13s 99ms/step - loss: 0.9203 -
categorical_accuracy: 0.7375 - val_loss: 1.0306 - val_categorical_accuracy:
0.7080
Epoch 35/100
128/128 [==============================] - 14s 102ms/step - loss: 0.9304 -
categorical_accuracy: 0.7280 - val_loss: 1.0271 - val_categorical_accuracy:
0.7041
Epoch 36/100
128/128 [==============================] - 13s 99ms/step - loss: 0.9243 -
categorical_accuracy: 0.7307 - val_loss: 1.0224 - val_categorical_accuracy:
0.7080
Epoch 37/100
128/128 [==============================] - 13s 100ms/step - loss: 0.9010 -
categorical_accuracy: 0.7432 - val_loss: 1.0179 - val_categorical_accuracy:
0.7100
Epoch 38/100
128/128 [==============================] - 14s 103ms/step - loss: 0.8840 -
```

```
categorical_accuracy: 0.7529 - val_loss: 1.0155 - val_categorical_accuracy:
0.7070
Epoch 39/100
128/128 [==============================] - 13s 101ms/step - loss: 0.8909 -
categorical_accuracy: 0.7402 - val_loss: 1.0122 - val_categorical_accuracy:
0.7090
Epoch 40/100
128/128 [==============================] - 14s 102ms/step - loss: 0.8772 -
categorical_accuracy: 0.7407 - val_loss: 1.0092 - val_categorical_accuracy:
0.7100
Epoch 41/100
128/128 [==============================] - 13s 101ms/step - loss: 0.8779 -
categorical_accuracy: 0.7454 - val_loss: 1.0073 - val_categorical_accuracy:
0.7100
Epoch 42/100
128/128 [==============================] - 13s 100ms/step - loss: 0.8661 -
categorical_accuracy: 0.7524 - val_loss: 1.0050 - val_categorical_accuracy:
0.7148
Epoch 43/100
128/128 [==============================] - 16s 120ms/step - loss: 0.8630 -
categorical_accuracy: 0.7542 - val_loss: 1.0029 - val_categorical_accuracy:
0.7139
Epoch 44/100
128/128 [==============================] - 13s 100ms/step - loss: 0.8588 -
categorical_accuracy: 0.7476 - val_loss: 0.9977 - val_categorical_accuracy:
0.7129
Epoch 45/100
128/128 [==============================] - 13s 100ms/step - loss: 0.8525 -
categorical_accuracy: 0.7542 - val_loss: 0.9980 - val_categorical_accuracy:
0.7148
Epoch 46/100
128/128 [==============================] - 13s 99ms/step - loss: 0.8541 -
categorical_accuracy: 0.7507 - val_loss: 0.9948 - val_categorical_accuracy:
0.7158
Epoch 47/100
128/128 [==============================] - 13s 101ms/step - loss: 0.8406 -
categorical_accuracy: 0.7600 - val_loss: 0.9924 - val_categorical_accuracy:
0.7168
Epoch 48/100
128/128 [==============================] - 14s 100ms/step - loss: 0.8372 -
categorical_accuracy: 0.7559 - val_loss: 0.9903 - val_categorical_accuracy:
0.7197
Epoch 49/100
128/128 [==============================] - 13s 100ms/step - loss: 0.8347 -
categorical_accuracy: 0.7600 - val_loss: 0.9878 - val_categorical_accuracy:
0.7217
Epoch 50/100
128/128 [==============================] - 13s 100ms/step - loss: 0.8284 -
```

```
categorical_accuracy: 0.7622 - val_loss: 0.9862 - val_categorical_accuracy:
0.7207
Epoch 51/100
128/128 [==============================] - 13s 98ms/step - loss: 0.8138 -
categorical_accuracy: 0.7656 - val_loss: 0.9850 - val_categorical_accuracy:
0.7197
Epoch 52/100
128/128 [==============================] - 13s 98ms/step - loss: 0.8219 -
categorical_accuracy: 0.7639 - val_loss: 0.9840 - val_categorical_accuracy:
0.7188
Epoch 53/100
128/128 [==============================] - 13s 99ms/step - loss: 0.8219 -
categorical_accuracy: 0.7620 - val_loss: 0.9819 - val_categorical_accuracy:
0.7227
Epoch 54/100
128/128 [==============================] - 13s 101ms/step - loss: 0.8203 -
categorical_accuracy: 0.7600 - val_loss: 0.9814 - val_categorical_accuracy:
0.7139
Epoch 55/100
128/128 [==============================] - 13s 99ms/step - loss: 0.8185 -
categorical_accuracy: 0.7561 - val_loss: 0.9799 - val_categorical_accuracy:
0.7197
Epoch 56/100
128/128 [==============================] - 13s 100ms/step - loss: 0.8034 -
categorical_accuracy: 0.7612 - val_loss: 0.9801 - val_categorical_accuracy:
0.7256
Epoch 57/100
128/128 [==============================] - 13s 101ms/step - loss: 0.7904 -
categorical_accuracy: 0.7654 - val_loss: 0.9772 - val_categorical_accuracy:
0.7246
Epoch 58/100
128/128 [==============================] - 13s 98ms/step - loss: 0.7979 -
categorical_accuracy: 0.7703 - val_loss: 0.9751 - val_categorical_accuracy:
0.7236
Epoch 59/100
128/128 [==============================] - 13s 97ms/step - loss: 0.7794 -
categorical_accuracy: 0.7786 - val_loss: 0.9761 - val_categorical_accuracy:
0.7246
Epoch 60/100
128/128 [==============================] - 13s 98ms/step - loss: 0.7936 -
categorical_accuracy: 0.7686 - val_loss: 0.9737 - val_categorical_accuracy:
0.7246
Epoch 61/100
128/128 [==============================] - 13s 97ms/step - loss: 0.7757 -
categorical_accuracy: 0.7700 - val_loss: 0.9732 - val_categorical_accuracy:
0.7256
Epoch 62/100
128/128 [==============================] - 13s 99ms/step - loss: 0.7876 -
```

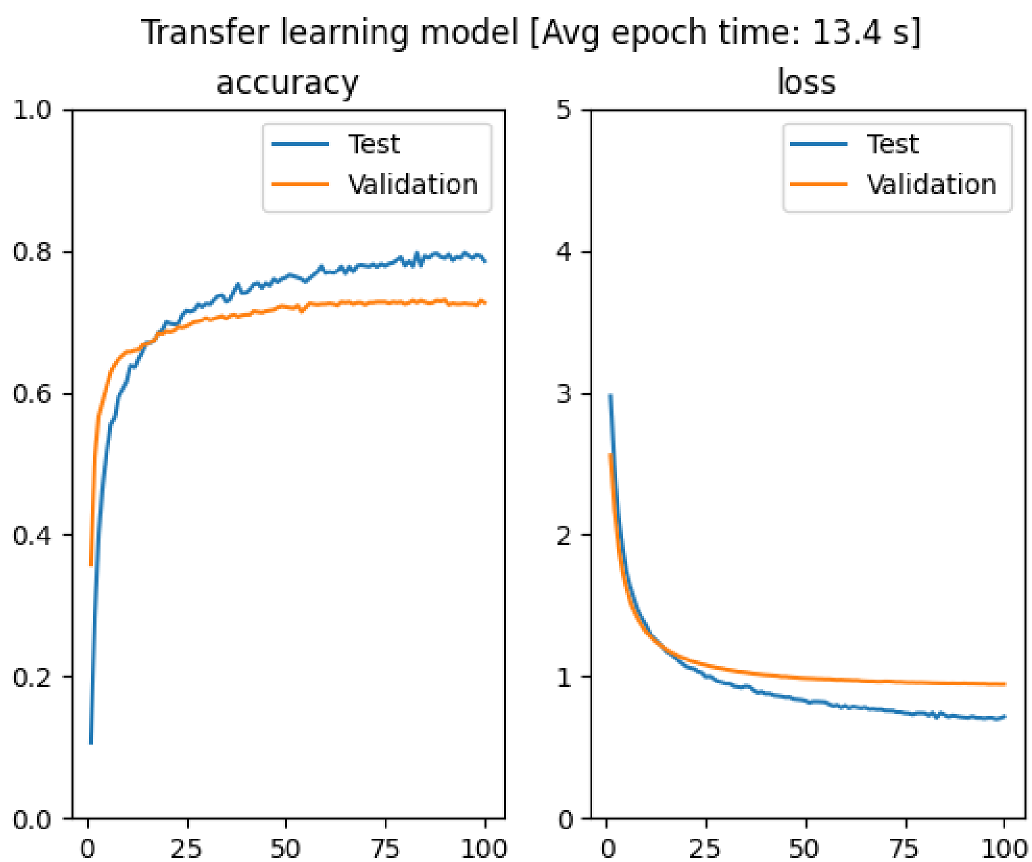
```
categorical_accuracy: 0.7688 - val_loss: 0.9711 - val_categorical_accuracy:
0.7246
Epoch 63/100
128/128 [==============================] - 13s 100ms/step - loss: 0.7815 -
categorical_accuracy: 0.7715 - val_loss: 0.9716 - val_categorical_accuracy:
0.7227
Epoch 64/100
128/128 [==============================] - 13s 98ms/step - loss: 0.7760 -
categorical_accuracy: 0.7778 - val_loss: 0.9674 - val_categorical_accuracy:
0.7275
Epoch 65/100
128/128 [==============================] - 13s 99ms/step - loss: 0.7822 -
categorical_accuracy: 0.7676 - val_loss: 0.9659 - val_categorical_accuracy:
0.7266
Epoch 66/100
128/128 [==============================] - 13s 100ms/step - loss: 0.7684 -
categorical_accuracy: 0.7778 - val_loss: 0.9645 - val_categorical_accuracy:
0.7275
Epoch 67/100
128/128 [==============================] - 13s 98ms/step - loss: 0.7712 -
categorical_accuracy: 0.7705 - val_loss: 0.9643 - val_categorical_accuracy:
0.7246
Epoch 68/100
128/128 [==============================] - 15s 115ms/step - loss: 0.7661 -
categorical_accuracy: 0.7788 - val_loss: 0.9619 - val_categorical_accuracy:
0.7266
Epoch 69/100
128/128 [==============================] - 13s 98ms/step - loss: 0.7672 -
categorical_accuracy: 0.7800 - val_loss: 0.9621 - val_categorical_accuracy:
0.7266
Epoch 70/100
128/128 [==============================] - 13s 98ms/step - loss: 0.7582 -
categorical_accuracy: 0.7788 - val_loss: 0.9642 - val_categorical_accuracy:
0.7236
Epoch 71/100
128/128 [==============================] - 13s 99ms/step - loss: 0.7572 -
categorical_accuracy: 0.7778 - val_loss: 0.9639 - val_categorical_accuracy:
0.7266
Epoch 72/100
128/128 [==============================] - 13s 99ms/step - loss: 0.7564 -
categorical_accuracy: 0.7810 - val_loss: 0.9618 - val_categorical_accuracy:
0.7266
Epoch 73/100
128/128 [==============================] - 13s 100ms/step - loss: 0.7471 -
categorical_accuracy: 0.7773 - val_loss: 0.9592 - val_categorical_accuracy:
0.7275
Epoch 74/100
128/128 [==============================] - 13s 99ms/step - loss: 0.7471 -
```

```
categorical_accuracy: 0.7812 - val_loss: 0.9598 - val_categorical_accuracy:
0.7275
Epoch 75/100
128/128 [==============================] - 13s 102ms/step - loss: 0.7396 -
categorical_accuracy: 0.7786 - val_loss: 0.9576 - val_categorical_accuracy:
0.7275
Epoch 76/100
128/128 [==============================] - 13s 100ms/step - loss: 0.7381 -
categorical_accuracy: 0.7817 - val_loss: 0.9563 - val_categorical_accuracy:
0.7256
Epoch 77/100
128/128 [==============================] - 13s 101ms/step - loss: 0.7300 -
categorical_accuracy: 0.7832 - val_loss: 0.9569 - val_categorical_accuracy:
0.7275
Epoch 78/100
128/128 [==============================] - 13s 99ms/step - loss: 0.7396 -
categorical_accuracy: 0.7871 - val_loss: 0.9554 - val_categorical_accuracy:
0.7266
Epoch 79/100
128/128 [==============================] - 13s 98ms/step - loss: 0.7401 -
categorical_accuracy: 0.7908 - val_loss: 0.9561 - val_categorical_accuracy:
0.7275
Epoch 80/100
128/128 [==============================] - 13s 98ms/step - loss: 0.7389 -
categorical_accuracy: 0.7793 - val_loss: 0.9551 - val_categorical_accuracy:
0.7246
Epoch 81/100
128/128 [==============================] - 13s 98ms/step - loss: 0.7228 -
categorical_accuracy: 0.7852 - val_loss: 0.9541 - val_categorical_accuracy:
0.7266
Epoch 82/100
128/128 [==============================] - 14s 102ms/step - loss: 0.7434 -
categorical_accuracy: 0.7781 - val_loss: 0.9529 - val_categorical_accuracy:
0.7295
Epoch 83/100
128/128 [==============================] - 13s 101ms/step - loss: 0.7067 -
categorical_accuracy: 0.7969 - val_loss: 0.9524 - val_categorical_accuracy:
0.7266
Epoch 84/100
128/128 [==============================] - 13s 101ms/step - loss: 0.7420 -
categorical_accuracy: 0.7788 - val_loss: 0.9516 - val_categorical_accuracy:
0.7295
Epoch 85/100
128/128 [==============================] - 13s 101ms/step - loss: 0.7274 -
categorical_accuracy: 0.7925 - val_loss: 0.9514 - val_categorical_accuracy:
0.7275
Epoch 86/100
128/128 [==============================] - 13s 99ms/step - loss: 0.7131 -
```

```
categorical_accuracy: 0.7900 - val_loss: 0.9501 - val_categorical_accuracy:
0.7256
Epoch 87/100
128/128 [==============================] - 13s 99ms/step - loss: 0.7230 -
categorical_accuracy: 0.7947 - val_loss: 0.9514 - val_categorical_accuracy:
0.7266
Epoch 88/100
128/128 [==============================] - 13s 98ms/step - loss: 0.7180 -
categorical_accuracy: 0.7959 - val_loss: 0.9500 - val_categorical_accuracy:
0.7295
Epoch 89/100
128/128 [==============================] - 13s 101ms/step - loss: 0.7124 -
categorical_accuracy: 0.7913 - val_loss: 0.9498 - val_categorical_accuracy:
0.7285
Epoch 90/100
128/128 [==============================] - 13s 101ms/step - loss: 0.7090 -
categorical_accuracy: 0.7896 - val_loss: 0.9512 - val_categorical_accuracy:
0.7305
Epoch 91/100
128/128 [==============================] - 14s 104ms/step - loss: 0.7062 -
categorical_accuracy: 0.7947 - val_loss: 0.9490 - val_categorical_accuracy:
0.7236
Epoch 92/100
128/128 [==============================] - 13s 101ms/step - loss: 0.7165 -
categorical_accuracy: 0.7866 - val_loss: 0.9494 - val_categorical_accuracy:
0.7256
Epoch 93/100
128/128 [==============================] - 14s 102ms/step - loss: 0.7062 -
categorical_accuracy: 0.7913 - val_loss: 0.9485 - val_categorical_accuracy:
0.7246
Epoch 94/100
128/128 [==============================] - 13s 102ms/step - loss: 0.7064 -
categorical_accuracy: 0.7898 - val_loss: 0.9475 - val_categorical_accuracy:
0.7246
Epoch 95/100
128/128 [==============================] - 13s 100ms/step - loss: 0.7024 -
categorical_accuracy: 0.7969 - val_loss: 0.9480 - val_categorical_accuracy:
0.7256
Epoch 96/100
128/128 [==============================] - 13s 99ms/step - loss: 0.7069 -
categorical_accuracy: 0.7930 - val_loss: 0.9446 - val_categorical_accuracy:
0.7246
Epoch 97/100
128/128 [==============================] - 13s 99ms/step - loss: 0.7066 -
categorical_accuracy: 0.7896 - val_loss: 0.9448 - val_categorical_accuracy:
0.7246
Epoch 98/100
128/128 [==============================] - 13s 100ms/step - loss: 0.6978 -
```

```
categorical_accuracy: 0.7939 - val_loss: 0.9453 - val_categorical_accuracy:
0.7227
Epoch 99/100
128/128 [==============================] - 14s 102ms/step - loss: 0.7041 -
categorical_accuracy: 0.7922 - val_loss: 0.9439 - val_categorical_accuracy:
0.7295
Epoch 100/100
128/128 [==============================] - 14s 102ms/step - loss: 0.7150 -
categorical_accuracy: 0.7854 - val_loss: 0.9444 - val_categorical_accuracy:
0.7266
```

```
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op while saving (showing
5 of 91). These functions will not be directly callable after loading.
```



[ ]:
```
transferred_model.evaluate(validation_ds)
```

```
32/32 [==============================] - 2s 75ms/step - loss: 0.9444 -
categorical_accuracy: 0.7266
```

[ ]: [0.944395899772644, 0.7265625]

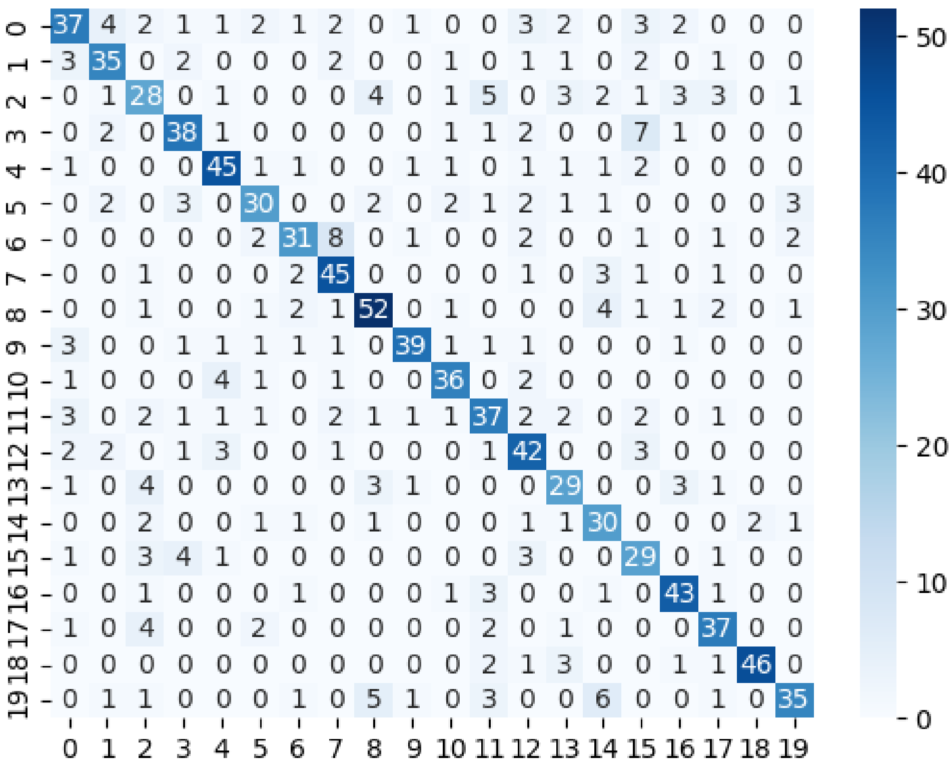
1.4.1 Tablica pomyłek

[ ]:
```
predictions = np.array([])
labels =  np.array([])
for x, y in validation_ds:
  predictions = np.concatenate([predictions, np.argmax(transferred_model.
 ↪predict(x), axis=-1)])
  labels = np.concatenate([labels, np.argmax(y, axis=-1)])

cf_matrix = tf.math.confusion_matrix(labels=labels, predictions=predictions).
 ↪numpy()
sns.heatmap(cf_matrix,
            annot=True,
            cmap='Blues')
```

```
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 52ms/step
1/1 [==============================] - 0s 70ms/step
1/1 [==============================] - 0s 79ms/step
1/1 [==============================] - 0s 57ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 58ms/step
1/1 [==============================] - 0s 49ms/step
1/1 [==============================] - 0s 48ms/step
1/1 [==============================] - 0s 58ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 49ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 52ms/step
1/1 [==============================] - 0s 56ms/step
1/1 [==============================] - 0s 67ms/step
1/1 [==============================] - 0s 60ms/step
1/1 [==============================] - 0s 56ms/step
1/1 [==============================] - 0s 55ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 62ms/step
1/1 [==============================] - 0s 49ms/step
1/1 [==============================] - 0s 53ms/step
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 42ms/step
```

```
1/1 [==============================] - 0s 38ms/step
```

[ ]: <Axes: >



Powyższa tabela pokazuje, że nasz model dosyć dobrze radzi sobie z zadaniem klasyfikacji. Najwięcej (dokładnie 8) błędów popełnia dla klas nr 6 i 7, czyli *ice cream* i *fish and chips*. Jest to jednak stosunkowo mała liczba w porównaniu z klasyfikacją poprawnych klas, która waha się w między 35 a ponad 50.

1.5 Fine tuning

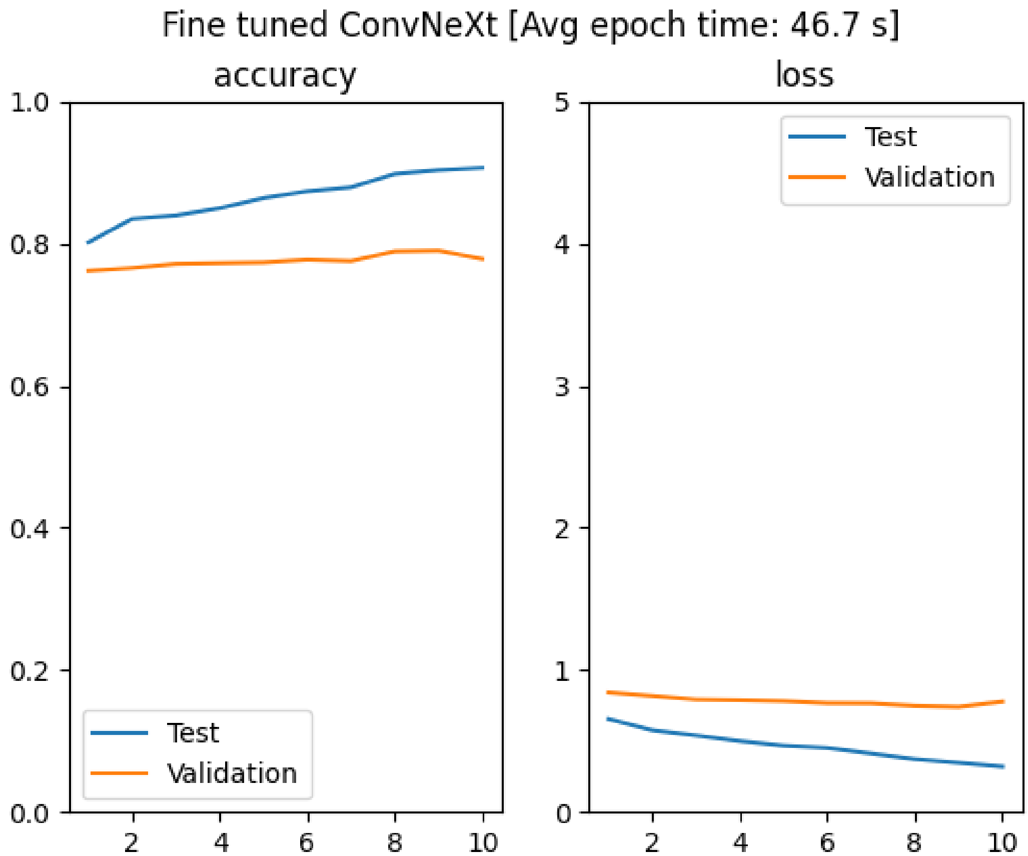
Następnie zastosowaliśmy fine-tuning, który pozwolił na zwiększenie precyzji do 77% (różnica 5 pp.).

[ ]:
```
def fine_tune(model_after_transfer_learning):
    base_model_idx = 2
    base_model = model_after_transfer_learning.get_layer(index=base_model_idx)
    base_model.trainable = True

    train_and_check_model(model_after_transfer_learning,
                          "Fine tuned ConvNeXt",
                          t_ds=train_ds,
                          v_ds=validation_ds,
```

```
                              epochs=10, lr_scale=0.01)
fine_tune(transferred_model)
```

```
Epoch 1/10
128/128 [==============================] - 131s 311ms/step - loss: 0.6556 -
categorical_accuracy: 0.8018 - val_loss: 0.8422 - val_categorical_accuracy:
0.7617
Epoch 2/10
128/128 [==============================] - 37s 287ms/step - loss: 0.5766 -
categorical_accuracy: 0.8347 - val_loss: 0.8179 - val_categorical_accuracy:
0.7656
Epoch 3/10
128/128 [==============================] - 37s 286ms/step - loss: 0.5398 -
categorical_accuracy: 0.8396 - val_loss: 0.7928 - val_categorical_accuracy:
0.7715
Epoch 4/10
128/128 [==============================] - 37s 285ms/step - loss: 0.5003 -
categorical_accuracy: 0.8499 - val_loss: 0.7888 - val_categorical_accuracy:
0.7725
Epoch 5/10
128/128 [==============================] - 37s 286ms/step - loss: 0.4679 -
categorical_accuracy: 0.8640 - val_loss: 0.7819 - val_categorical_accuracy:
0.7734
Epoch 6/10
128/128 [==============================] - 37s 286ms/step - loss: 0.4522 -
categorical_accuracy: 0.8735 - val_loss: 0.7674 - val_categorical_accuracy:
0.7773
Epoch 7/10
128/128 [==============================] - 37s 284ms/step - loss: 0.4128 -
categorical_accuracy: 0.8792 - val_loss: 0.7661 - val_categorical_accuracy:
0.7754
Epoch 8/10
128/128 [==============================] - 38s 292ms/step - loss: 0.3731 -
categorical_accuracy: 0.8979 - val_loss: 0.7487 - val_categorical_accuracy:
0.7891
Epoch 9/10
128/128 [==============================] - 38s 292ms/step - loss: 0.3479 -
categorical_accuracy: 0.9033 - val_loss: 0.7416 - val_categorical_accuracy:
0.7900
Epoch 10/10
128/128 [==============================] - 38s 292ms/step - loss: 0.3214 -
categorical_accuracy: 0.9065 - val_loss: 0.7778 - val_categorical_accuracy:
0.7783
```

```
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op while saving (showing
5 of 91). These functions will not be directly callable after loading.
```

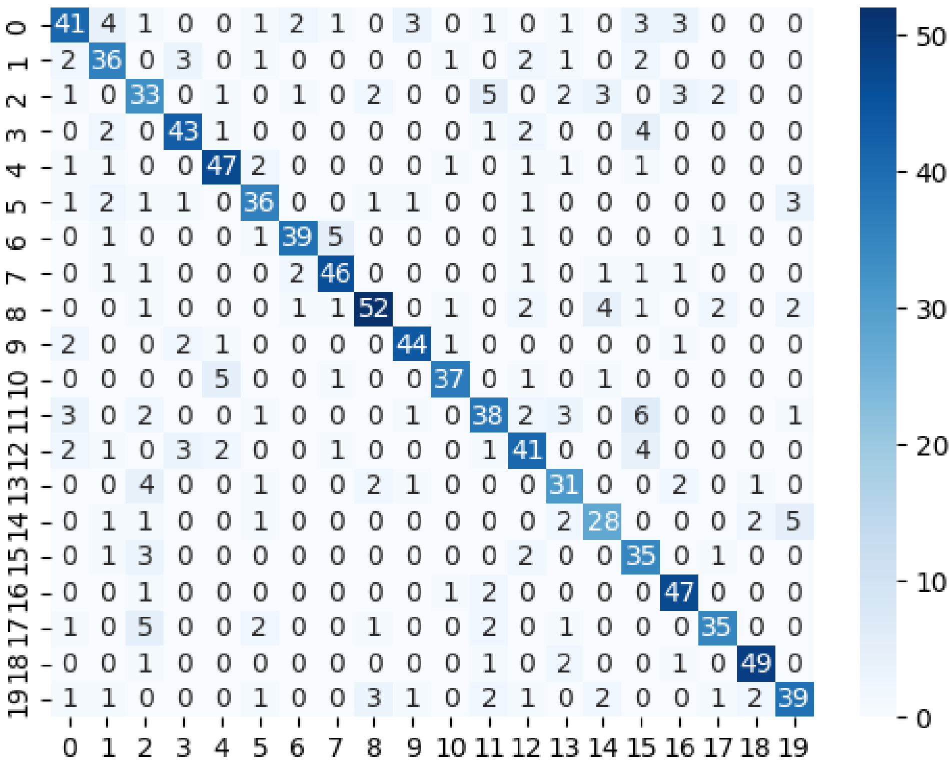



```
[ ]: transferred_model.evaluate(validation_ds)

predictions = np.array([])
labels =  np.array([])
for x, y in validation_ds:
  predictions = np.concatenate([predictions, np.argmax(transferred_model.
 ↪predict(x), axis=-1)])
  labels = np.concatenate([labels, np.argmax(y, axis=-1)])

cf_matrix = tf.math.confusion_matrix(labels=labels, predictions=predictions).
 ↪numpy()
sns.heatmap(cf_matrix,
            annot=True,
            cmap='Blues')
```

```
32/32 [==============================] - 3s 81ms/step - loss: 0.7778 -
categorical_accuracy: 0.7783
1/1 [==============================] - 2s 2s/step
1/1 [==============================] - 0s 66ms/step
1/1 [==============================] - 0s 88ms/step
```

```
1/1 [==============================] - 0s 70ms/step
1/1 [==============================] - 0s 52ms/step
1/1 [==============================] - 0s 57ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 62ms/step
1/1 [==============================] - 0s 54ms/step
1/1 [==============================] - 0s 56ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 58ms/step
1/1 [==============================] - 0s 61ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 54ms/step
1/1 [==============================] - 0s 82ms/step
1/1 [==============================] - 0s 103ms/step
1/1 [==============================] - 0s 67ms/step
1/1 [==============================] - 0s 80ms/step
1/1 [==============================] - 0s 77ms/step
1/1 [==============================] - 0s 146ms/step
1/1 [==============================] - 0s 71ms/step
1/1 [==============================] - 0s 96ms/step
1/1 [==============================] - 0s 54ms/step
1/1 [==============================] - 0s 65ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 39ms/step
```

```
[ ]: <Axes: >
```



Jak należało się spodziewać liczba poprawnych wyników zwiększyła się. Wspomniana wcześniej liczba błędów dla problemu klasyfikacji klas nr 6 i 7 zmalała z 8 do 5. Model znacznie lepiej radzi sobie z trudniejszymi przykładami niż model, na którym nie zastosowano fine-tuningu.

1.6 Wnioski

Transfer learning pozwala na szybkie uczenie, co pozwala zaoszczędzić czas i zasoby. Dzieje się tak, ponieważ wykorzystuje wstępnie wytrenowane modele, które nauczyły się reprezentacji danych na dużych zbiorach danych. Przygotowany w ten sposób model radzi sobie zdecydowanie lepiej niż wytrenowany od zera. Jest to szczególnie istotne w przypadku ograniczonej ilości dostępnych danych. Wyniki powyżej wytrenowanych modeli potwierdzają te wnioski.

Z kolei fine-tuning pozwala na aktualizację parametrów modelu na nowych danych, co umożliwia lepszą adaptację modelu do specyficznych warunków i zmian w danych wejściowych. W niniejszej pracy zastosowanie tej techniki pozwoliło znacząco polepszyć wyniki modelu.