Практическое задание №1

В настоящее время методы глубокого обучения показывают высокие достижения в классификации, сегментации и анализе биомедицинских изображений, включая гистологические изображения.

В данном исследовании основное внимание уделяется задаче классификации отдельных участков гистологических тканей.

Подготовка

Установка необходимых пакетов:

In [1]:

```
!pip install -q tqdm
!pip install --upgrade --no-cache-dir gdown
```

```
Requirement already satisfied: gdown in /usr/local/lib/python3.10/dist-packages (4.6.6)
Collecting gdown
  Downloading gdown-4.7.1-py3-none-any.whl (15 kB)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from
gdown) (3.13.1)
Requirement already satisfied: requests[socks] in /usr/local/lib/python3.10/dist-packages
(from gdown) (2.31.0)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from gdown
) (1.16.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from gdow
n) (4.66.1)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-packages
(from gdown) (4.11.2)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-packages (
from beautifulsoup4->gdown) (2.5)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist
-packages (from requests[socks]->gdown) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (f
rom requests[socks]->gdown) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packa
ges (from requests[socks]->gdown) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packa
ges (from requests[socks]->gdown) (2023.7.22)
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in /usr/local/lib/python3.10/dist-p
ackages (from requests[socks]->gdown) (1.7.1)
Installing collected packages: gdown
  Attempting uninstall: gdown
    Found existing installation: gdown 4.6.6
    Uninstalling gdown-4.6.6:
      Successfully uninstalled gdown-4.6.6
Successfully installed gdown-4.7.1
```

Монтирование Вашего **Google Drive** к текущему окружению:

In [2]:

```
from google.colab import drive
drive.mount('/content/drive', force_remount=True)
```

```
Mounted at /content/drive
```

Константы, которые пригодятся в коде далее, и ссылки **(gdrive** идентификаторы**)** на предоставляемые наборы данных:

In [3]:

```
EVALUATE_ONLY = True
TEST_ON_LARGE_DATASET = True
TISSUE_CLASSES = ('ADI', 'BACK', 'DEB', 'LYM', 'MUC', 'MUS', 'NORM', 'STR', 'TUM')
DATASETS_LINKS = {
    'train': '1XtQzVQ5XbrfxpLHJuL0XBGJ5U7CS-cLi',
    'train_small': '1qd45xXfDwdZjktLFwQb-et-mAaFeCzOR',
    'train_tiny': '1I-2ZOuXLd4QwhZQQltp817Kn3J0Xgbui',
    'test': '1RfPou3pFKpuHDJZ-D9XDFzgvwpUBFlDr',
    'test_small': '1wbRsog0n7uGlHIPGLhyN-PMeT2kdQ2lI',
    'test_tiny': '1viiB0s041CNsAK4itvX8PnYthJ-MDnQc'
}
```

Импорт необходимых зависимостей:

In [4]:

```
from pathlib import Path
import numpy as np
from typing import List
from tqdm.notebook import tqdm
from time import sleep
from PIL import Image
import IPython.display
from sklearn.metrics import balanced_accuracy_score
import gdown
import os
from tensorflow.keras.applications import ResNet101V2
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.optimizers import Adam
```

Класс **Dataset**

Предназначен для работы с наборами данных, обеспечивает чтение изображений и соответствующих меток, а также формирование пакетов (батчей).

Замечание: Пришлось внести изменение в класс **Dataset**, так как в входе проверки класса, столкнулся с проблемой **- Access denied with the following error: Loading Dataset train_small from npz.**

Поэтому, возможно если Вы столкнулись с той же проблемой - раскомментируйте блок кода ниже и закомментируйте другой. Оба варианта работают исправно.

> *Пожалуйста, не забудьте что нужно указать другой путь до набора данных. У меня находится в папке **My Drive/Colab Notebooks***

In [5]:

```
class Dataset:

    def __init__(self, name):
        self.name = name
        self.is_loaded = False

        # Uncomment. If you have got rid of the problem. "Don't forget to comment below t
he block of code."
        ''' url = f"https://drive.google.com/uc?export=download&confirm=pbef&id={DATASETS
_LINKS[name]}"
        output = f'{name}.npz'
        gdown.download(url, output, quiet=False)
        print(f'Loading dataset {self.name} from npz.')
        np_obj = np.load(f'{name}.npz') '''

        # Get the path to the current directory where the trained date set is located
        current_directory = '/content/drive/My Drive/Colab Notebooks'
```

```
        file_path = os.path.join(current_directory, f"{name}.npz")
        # Load the data from the file
        np_obj = np.load(file_path)

        self.images = np_obj['data']
        self.labels = np_obj['labels']
        self.n_files = self.images.shape[0]
        self.is_loaded = True
        print(f'Done. Dataset {name} consists of {self.n_files} images.')

    def image(self, i):
        # read i-th image in dataset and return it as numpy array
        if self.is_loaded:
            return self.images[i, :, :, :]

    def images_seq(self, n=None):
        # sequential access to images inside dataset (is needed for testing)
        for i in range(self.n_files if not n else n):
            yield self.image(i)

    def random_image_with_label(self):
        # get random image with label from dataset
        i = np.random.randint(self.n_files)
        return self.image(i), self.labels[i]

    def random_batch_with_labels(self, n):
        # create random batch of images with labels (is needed for training)
        indices = np.random.choice(self.n_files, n)
        imgs = []
        for i in indices:
            img = self.image(i)
            imgs.append(self.image(i))
        logits = np.array([self.labels[i] for i in indices])
        return np.stack(imgs), logits

    def image_with_label(self, i: int):
        # return i-th image with label from dataset
        return self.image(i), self.labels[i]
```

Пример испольвания класса **Dataset**

Загрузим обучающий набор данных, получим произвольное изображение с меткой. После чего визуализируем изображение, выведем метку. Этот фрагмент кода не несет никакой информационной ценности, а лишь предназначен для проверки корректности выполнения класса **Dataset."**

In [8]:

```
d_train_tiny = Dataset('train_small')

img, lbl = d_train_tiny.random_image_with_label()
print()
print(f'Got numpy array of shape {img.shape}, and label with code {lbl}.')
print(f'Label code corresponds to {TISSUE_CLASSES[lbl]} class.')

pil_img = Image.fromarray(img)
IPython.display.display(pil_img)
```

```
Done. Dataset train_small consists of 7200 images.

Got numpy array of shape (224, 224, 3), and label with code 0.
Label code corresponds to ADI class.
```

Класс **Metrics**

Реализует метрики точности, используемые для оценивания модели:

1. точность,
2. сбалансированную точность.

In [6]:

```
class Metrics:

    @staticmethod
    def accuracy(gt: List[int], pred: List[int]):
        assert len(gt) == len(pred), 'gt and prediction should be of equal length'
        return sum(int(i[0] == i[1]) for i in zip(gt, pred)) / len(gt)

    @staticmethod
    def accuracy_balanced(gt: List[int], pred: List[int]):
        return balanced_accuracy_score(gt, pred)

    @staticmethod
    def print_all(gt: List[int], pred: List[int], info: str):
        print(f'metrics for {info}:')
        print('\t accuracy {:.4f}:'.format(Metrics.accuracy(gt, pred)))
        print('\t balanced accuracy {:.4f}:'.format(Metrics.accuracy_balanced(gt, pred))
)
```

Класс **Model**

Класс, хранящий в себе всю информацию о модели.

In [7]:

```
class Model:

    def __init__(self, input_shape=(224, 224, 3), num_classes=9):
        self.model = self.build_model(input_shape, num_classes)

    def build_model(self, input_shape, num_classes):
        #LBL13
        base_model = ResNet101V2(weights='imagenet', include_top=False, input_shape=inpu
t_shape)
        model = Sequential()
        model.add(base_model)
        model.add(GlobalAveragePooling2D())
        model.add(Dense(num_classes, activation='softmax'))
        return model

    def save(self, name: str):
        self.model.save(f'{name}.h5')

    def load(self, name: str):
        DATASETS_LINKS = {
            'best_final': '1-7AICbd8zpHrSZHwcstv8DqZa4TEixlg',
            'best_small':'1-1hmjHOaxJ29SHmbs2Pf8zLl4jy4SaBB',
            'best_tiny': '1qb9BK5TAqJak9QlRTrsKaM7IAgT6Lgm-'
        }
        link = f"https://drive.google.com/uc?export=download&id={DATASETS_LINKS.get(name,
```

```
''')}"
        gdown.download(link, f'{name}.h5', quiet=False)
        self.model.load_weights(f'{name}.h5')

    def train(self, dataset: Dataset, epochs=10, batch_size=32):
        self.model.compile(optimizer=Adam(), loss='sparse_categorical_crossentropy', metr
ics=['accuracy'])
        self.model.fit(dataset.images, dataset.labels, epochs=epochs, batch_size=batch_s
ize)

    def test_on_dataset(self, dataset: Dataset, limit=None):
        predictions = []
        n = dataset.n_files if not limit else int(dataset.n_files * limit)
        for i in tqdm(range(n)):
            img, label = dataset.image_with_label(i)
            predictions.append(self.test_on_image(img))
        return predictions

    def test_on_image(self, img: np.ndarray):
        prediction = self.model.predict(np.expand_dims(img, axis=0))[0]
        return np.argmax(prediction)
```

Классификация изображений

Используя введенные выше классы можем перейти уже непосредственно к обучению модели классификации изображений. Пример общего пайплайна решения задачи приведен ниже.

In [ ]:

```
model = Model()
```

Обучение на *train_tiny*

In [ ]:

```
d_train_tiny = Dataset('train_tiny')
model.train(d_train_tiny)
#LBL3
model.save('/content/drive/My Drive/Colab Notebooks/best_tiny')

#LBL5
```

```
Done. Dataset train_tiny consists of 900 images.
Epoch 1/10
29/29 [==============================] - 88s 486ms/step - loss: 1.1205 - accuracy: 0.6444
Epoch 2/10
29/29 [==============================] - 13s 441ms/step - loss: 0.7243 - accuracy: 0.7611
Epoch 3/10
29/29 [==============================] - 13s 441ms/step - loss: 0.7540 - accuracy: 0.7678
Epoch 4/10
29/29 [==============================] - 13s 447ms/step - loss: 0.6919 - accuracy: 0.7844
Epoch 5/10
29/29 [==============================] - 13s 448ms/step - loss: 0.5348 - accuracy: 0.8289
Epoch 6/10
29/29 [==============================] - 13s 451ms/step - loss: 0.4145 - accuracy: 0.8667
Epoch 7/10
29/29 [==============================] - 13s 454ms/step - loss: 0.3553 - accuracy: 0.8911
Epoch 8/10
29/29 [==============================] - 13s 458ms/step - loss: 0.3339 - accuracy: 0.8967
Epoch 9/10
29/29 [==============================] - 14s 469ms/step - loss: 0.2664 - accuracy: 0.8978
Epoch 10/10
29/29 [==============================] - 14s 472ms/step - loss: 0.2223 - accuracy: 0.9300
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3079: UserWarning: Y
ou are saving your model as an HDF5 file via `model.save()`. This file format is consider
ed legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model
```

```
.keras') .
  saving_api.save_model(
```

Обучение на *train_small*

In [ ]:

```
#LBL4
d_train_small = Dataset('train_small')
model.load('/content/drive/My Drive/Colab Notebooks/best_tiny')  # Loading the weights fr
om the previous step
model.train(d_train_small)
model.save('/content/drive/My Drive/Colab Notebooks/best_small')
```

```
Done. Dataset train_small consists of 7200 images.
Epoch 1/10
225/225 [==============================] - 164s 465ms/step - loss: 0.4982 - accuracy: 0.8
431
Epoch 2/10
225/225 [==============================] - 107s 476ms/step - loss: 0.2848 - accuracy: 0.9
078
Epoch 3/10
225/225 [==============================] - 109s 483ms/step - loss: 0.2175 - accuracy: 0.9
267
Epoch 4/10
225/225 [==============================] - 109s 485ms/step - loss: 0.1525 - accuracy: 0.9
476
Epoch 5/10
225/225 [==============================] - 111s 491ms/step - loss: 0.1421 - accuracy: 0.9
526
Epoch 6/10
225/225 [==============================] - 110s 490ms/step - loss: 0.1101 - accuracy: 0.9
611
Epoch 7/10
225/225 [==============================] - 109s 487ms/step - loss: 0.0992 - accuracy: 0.9
654
Epoch 8/10
225/225 [==============================] - 109s 486ms/step - loss: 0.1017 - accuracy: 0.9
657
Epoch 9/10
225/225 [==============================] - 109s 486ms/step - loss: 0.0652 - accuracy: 0.9
786
Epoch 10/10
225/225 [==============================] - 109s 485ms/step - loss: 0.0662 - accuracy: 0.9
775
```

Обучение на *train*

In [ ]:

```
d_train = Dataset('train')
model.load('/content/drive/My Drive/Colab Notebooks/best_small') # Loading the weights fr
om the previous step
model.train(d_train)
model.save('/content/drive/My Drive/Colab Notebooks/best_final')
```

```
Done. Dataset train consists of 18000 images.
Epoch 1/10
563/563 [==============================] - 345s 491ms/step - loss: 0.1405 - accuracy: 0.9
539
Epoch 2/10
563/563 [==============================] - 275s 488ms/step - loss: 0.1123 - accuracy: 0.9
634
Epoch 3/10
563/563 [==============================] - 275s 489ms/step - loss: 0.0849 - accuracy: 0.9
716
Epoch 4/10
563/563 [==============================] - 275s 489ms/step - loss: 0.0837 - accuracy: 0.9
739
Epoch 5/10
```

```
563/563 [==============================] - 275s 488ms/step - loss: 0.0575 - accuracy: 0.9
806
Epoch 6/10
563/563 [==============================] - 275s 488ms/step - loss: 0.0561 - accuracy: 0.9
816
Epoch 7/10
563/563 [==============================] - 275s 488ms/step - loss: 0.0499 - accuracy: 0.9
829
Epoch 8/10
563/563 [==============================] - 275s 489ms/step - loss: 0.0423 - accuracy: 0.9
851
Epoch 9/10
563/563 [==============================] - 275s 489ms/step - loss: 0.0353 - accuracy: 0.9
879
Epoch 10/10
563/563 [==============================] - 275s 488ms/step - loss: 0.0474 - accuracy: 0.9
843
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3079: UserWarning: Y
ou are saving your model as an HDF5 file via `model.save()`. This file format is consider
ed legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model
.keras')`.
  saving_api.save_model(
```

Пример тестирования модели на части набора данных:

In [ ]:

```
d_test = Dataset('test')
# evaluating model on 10% of test dataset
pred_1 = model.test_on_dataset(d_test, limit=0.1)
Metrics.print_all(d_test.labels[:len(pred_1)], pred_1, '10% of test')
```

```
Done. Dataset test consists of 4500 images.


1/1 [==============================] - 0s 55ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 32ms/step
```

```
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 31ms/step
metrics for 10% of test:
  accuracy 0.9533:
  balanced accuracy 0.9533:
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:2184: UserWarn
ing: y_pred contains classes not in y_true
  warnings.warn("y_pred contains classes not in y_true")
```

Пример тестирования модели на полном наборе данных:

In [ ]:

```
# evaluating model on full test dataset (may take time)
if TEST_ON_LARGE_DATASET:
    pred_2 = model.test_on_dataset(d_test)
    Metrics.print_all(d_test.labels, pred_2, 'test')
```

```
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 63ms/step
1/1 [==============================] - 0s 90ms/step
1/1 [==============================] - 0s 69ms/step
1/1 [==============================] - 0s 60ms/step
1/1 [==============================] - 0s 57ms/step
1/1 [==============================] - 0s 61ms/step
1/1 [==============================] - 0s 68ms/step
1/1 [==============================] - 0s 58ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 63ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 72ms/step
1/1 [==============================] - 0s 63ms/step
1/1 [==============================] - 0s 48ms/step
1/1 [==============================] - 0s 92ms/step
1/1 [==============================] - 0s 61ms/step
1/1 [==============================] - 0s 54ms/step
1/1 [==============================] - 0s 60ms/step
1/1 [==============================] - 0s 79ms/step
1/1 [==============================] - 0s 57ms/step
1/1 [==============================] - 0s 63ms/step
1/1 [==============================] - 0s 60ms/step
1/1 [==============================] - 0s 73ms/step
1/1 [==============================] - 0s 69ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 66ms/step
1/1 [==============================] - 0s 65ms/step
1/1 [==============================] - 0s 52ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 44ms/step
1/1 [==============================] - 0s 48ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 48ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 83ms/step
1/1 [==============================] - 0s 42ms/step
metrics for test:
  accuracy 0.7876:
  balanced accuracy 0.7876:
```

Тестирование модели на других наборах данных

Модель должна поддерживать тестирование на других наборах данных. Ниже приведен фрагмент кода, который будет осуществлять тестирование для оценивания Вашей модели на дополнительных тестовых наборах данных.

In [8]:

```
final_model = Model()
final_model.load('best_final')
d_test_tiny = Dataset('test_tiny')
pred = final_model.test_on_dataset(d_test_tiny)
Metrics.print_all(d_test_tiny.labels, pred, 'test-tiny')
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet
/resnet101v2_weights_tf_dim_ordering_tf_kernels_notop.h5
171317808/171317808 [==============================] - 2s 0us/step

Downloading...
From (uriginal): https://drive.google.com/uc?export=download&id=1-7AICbd8zpHrSZHwcstv8DqZ
a4TEixlg
From (redirected): https://drive.google.com/uc?export=download&id=1-7AICbd8zpHrSZHwcstv8D
qZa4TEixlg&confirm=t&uuid=e0dcd667-24b5-4617-8d89-9aafc1717522
To: /content/best_final.h5
100%|██████████| 512M/512M [00:03<00:00, 134MB/s]

Done. Dataset test_tiny consists of 90 images.


1/1 [==============================] - 4s 4s/step
```

```
1/1 [==============================] - 0s 323ms/step
1/1 [==============================] - 0s 318ms/step
1/1 [==============================] - 1s 573ms/step
1/1 [==============================] - 0s 310ms/step
1/1 [==============================] - 1s 581ms/step
1/1 [==============================] - 0s 319ms/step
1/1 [==============================] - 0s 309ms/step
1/1 [==============================] - 0s 312ms/step
1/1 [==============================] - 0s 326ms/step
1/1 [==============================] - 0s 312ms/step
1/1 [==============================] - 0s 334ms/step
1/1 [==============================] - 0s 318ms/step
1/1 [==============================] - 0s 322ms/step
1/1 [==============================] - 0s 323ms/step
1/1 [==============================] - 0s 318ms/step
1/1 [==============================] - 0s 317ms/step
1/1 [==============================] - 0s 315ms/step
metrics for test-tiny:
  accuracy 0.7556:
  balanced accuracy 0.7556:
```

In [9]:

```
final_model = Model()
final_model.load('best_final')
d_test_tiny = Dataset('test_small')
pred = final_model.test_on_dataset(d_test_tiny)
Metrics.print_all(d_test_tiny.labels, pred, 'test-small')
```

```
Downloading...
From (uriginal): https://drive.google.com/uc?export=download&id=1-7AICbd8zpHrSZHwcstv8DqZa4TEixlg
From (redirected): https://drive.google.com/uc?export=download&id=1-7AICbd8zpHrSZHwcstv8DqZa4TEixlg&confirm=t&uuid=7693b92f-bf5b-4c33-b913-ec1096e862c5
To: /content/best_final.h5
100%|██████████| 512M/512M [00:02<00:00, 226MB/s]
```

```
Done. Dataset test_small consists of 1800 images.

1/1 [==============================] - 5s 5s/step
1/1 [==============================] - 1s 735ms/step
1/1 [==============================] - 1s 624ms/step
1/1 [==============================] - 0s 319ms/step
1/1 [==============================] - 0s 326ms/step
1/1 [==============================] - 0s 331ms/step
1/1 [==============================] - 0s 319ms/step
1/1 [==============================] - 0s 323ms/step
1/1 [==============================] - 0s 330ms/step
1/1 [==============================] - 0s 318ms/step
1/1 [==============================] - 0s 321ms/step
1/1 [==============================] - 0s 314ms/step
1/1 [==============================] - 0s 327ms/step
1/1 [==============================] - 1s 521ms/step
1/1 [==============================] - 1s 580ms/step
1/1 [==============================] - 1s 611ms/step
1/1 [==============================] - 1s 517ms/step
1/1 [==============================] - 0s 345ms/step
1/1 [==============================] - 0s 326ms/step
1/1 [==============================] - 0s 317ms/step
1/1 [==============================] - 1s 517ms/step
1/1 [==============================] - 0s 496ms/step
1/1 [==============================] - 1s 536ms/step
1/1 [==============================] - 1s 511ms/step
1/1 [==============================] - 1s 522ms/step
1/1 [==============================] - 1s 547ms/step
1/1 [==============================] - 1s 516ms/step
1/1 [==============================] - 1s 567ms/step
1/1 [==============================] - 1s 533ms/step
1/1 [==============================] - 1s 523ms/step
1/1 [==============================] - 0s 396ms/step
1/1 [==============================] - 0s 317ms/step
1/1 [==============================] - 0s 321ms/step
```

```
1/1 [==============================] - 1s 531ms/step
1/1 [==============================] - 1s 614ms/step
1/1 [==============================] - 1s 594ms/step
1/1 [==============================] - 1s 568ms/step
1/1 [==============================] - 1s 606ms/step
1/1 [==============================] - 1s 689ms/step
1/1 [==============================] - 1s 548ms/step
1/1 [==============================] - 1s 565ms/step
1/1 [==============================] - 1s 564ms/step
1/1 [==============================] - 1s 580ms/step
1/1 [==============================] - 1s 584ms/step
1/1 [==============================] - 1s 575ms/step
1/1 [==============================] - 1s 559ms/step
1/1 [==============================] - 1s 575ms/step
1/1 [==============================] - 1s 523ms/step
1/1 [==============================] - 1s 529ms/step
1/1 [==============================] - 0s 336ms/step
1/1 [==============================] - 0s 316ms/step
1/1 [==============================] - 0s 327ms/step
1/1 [==============================] - 0s 310ms/step
1/1 [==============================] - 0s 312ms/step
1/1 [==============================] - 0s 330ms/step
1/1 [==============================] - 0s 312ms/step
1/1 [==============================] - 0s 311ms/step
1/1 [==============================] - 0s 321ms/step
1/1 [==============================] - 0s 319ms/step
1/1 [==============================] - 0s 329ms/step
1/1 [==============================] - 0s 335ms/step
1/1 [==============================] - 0s 335ms/step
1/1 [==============================] - 0s 322ms/step
1/1 [==============================] - 0s 332ms/step
1/1 [==============================] - 0s 316ms/step
1/1 [==============================] - 0s 314ms/step
1/1 [==============================] - 0s 317ms/step
1/1 [==============================] - 0s 327ms/step
1/1 [==============================] - 0s 322ms/step
1/1 [==============================] - 0s 331ms/step
1/1 [==============================] - 0s 437ms/step
1/1 [==============================] - 1s 568ms/step
metrics for test-small:
  accuracy 0.7950:
  balanced accuracy 0.7950:
```

In [10]:

```
final_model = Model()
final_model.load('best_final')
d_test_tiny = Dataset('test')
pred = final_model.test_on_dataset(d_test_tiny)
Metrics.print_all(d_test_tiny.labels, pred, 'test')
```

```
Downloading...
From (uriginal): https://drive.google.com/uc?export=download&id=1-7AICbd8zpHrSZHwcstv8DqZa4TEixlg
From (redirected): https://drive.google.com/uc?export=download&id=1-7AICbd8zpHrSZHwcstv8DqZa4TEixlg&confirm=t&uuid=f5510021-b6e5-4dd0-90e0-75e85bb8db59
To: /content/best_final.h5
100%|██████████| 512M/512M [00:02<00:00, 199MB/s]
```

```
Done. Dataset test consists of 4500 images.


1/1 [==============================] - 3s 3s/step
1/1 [==============================] - 0s 337ms/step
1/1 [==============================] - 0s 330ms/step
1/1 [==============================] - 0s 340ms/step
1/1 [==============================] - 0s 321ms/step
1/1 [==============================] - 0s 315ms/step
1/1 [==============================] - 0s 317ms/step
1/1 [==============================] - 0s 323ms/step
1/1 [==============================] - 0s 318ms/step
1/1 [==============================] - 0s 318ms/step
1/1 [==============================] - 0s 325ms/step
```

```
1/1 [==============================] - 0s 310ms/step
1/1 [==============================] - 0s 329ms/step
1/1 [==============================] - 0s 314ms/step
1/1 [==============================] - 0s 334ms/step
1/1 [==============================] - 0s 321ms/step
1/1 [==============================] - 0s 344ms/step
1/1 [==============================] - 0s 322ms/step
1/1 [==============================] - 0s 339ms/step
1/1 [==============================] - 0s 319ms/step
1/1 [==============================] - 0s 325ms/step
1/1 [==============================] - 0s 320ms/step
1/1 [==============================] - 0s 322ms/step
1/1 [==============================] - 0s 354ms/step
1/1 [==============================] - 0s 336ms/step
1/1 [==============================] - 0s 362ms/step
1/1 [==============================] - 1s 568ms/step
1/1 [==============================] - 1s 542ms/step
1/1 [==============================] - 1s 590ms/step
1/1 [==============================] - 1s 523ms/step
1/1 [==============================] - 1s 535ms/step
1/1 [==============================] - 1s 555ms/step
1/1 [==============================] - 1s 550ms/step
1/1 [==============================] - 1s 559ms/step
1/1 [==============================] - 1s 536ms/step
1/1 [==============================] - 1s 550ms/step
metrics for test:
  accuracy 0.7876:
  balanced accuracy 0.7876:
```

Отмонтировать Google Drive.

In [ ]:

```
drive.flush_and_unmount()
```