

✓ Практическое задание №1

Установка необходимых пакетов:

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

```
Requirement already satisfied: gdown in /usr/local/lib/python3.10/dist-packages (5.2.0)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-packages (from gdown) (4.12.3)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from gdown) (3.16.1)
Requirement already satisfied: requests[socks] in /usr/local/lib/python3.10/dist-packages (from gdown) (2.32.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from gdown) (4.66.6)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-packages (from beautifulsoup4->gdown) (2.6)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (3)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (2024.8.3)
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (1.7
```

Монтирование Вашего Google Drive к текущему окружению:

```
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:

```
EVALUATE_ONLY = True
TEST_ON_LARGE_DATASET = True
TISSUE_CLASSES = ('ADI', 'BACK', 'DEB', 'LYM', 'MUC', 'MUS', 'NORM', 'STR', 'TUM')
DATASETS_LINKS = {
    'train': '1jmFzZRPv7CzmUUMziRQo4VTfbYJD8DA8',
    'train_small': '1nWFRMVUFAPBNqjwcc9iUZfSD-rC0X2N',
    'train_tiny': '1I-2Z0uXLd4QwhZQQ1tp817Kn3J0Xgbui',
    'test': '1V6jaw1sFnow64Z9QM6upcxtfLJ9KuUA',
    'test_small': '1kVxHCOK0fHFLzo4bK3nQncHzcmbem3IQ',
    'test_tiny': '1viiB0s041CNSAK4itvX8PnYthJ-MDnQc'
}
```

Импорт необходимых зависимостей:

```
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
from tensorflow.keras.applications import EfficientNetB0
from tensorflow.keras import layers, models
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator, random_rotation, random_shear, random_shift, random_zoom
from tensorflow.keras.callbacks import TensorBoard, ReduceLROnPlateau, EarlyStopping
import os
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
```

✓ Класс Dataset

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

```
class Dataset:
```

```

def __init__(self, name):
    self.name = name
    self.is_loaded = False
    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')
    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 использования класса Dataset

Загрузим обучающий набор данных, получим произвольное изображение с меткой. После чего визуализируем изображение, выведем метку. В будущем, этот кусок кода можно закомментировать или убрать.

```

d_train_tiny = Dataset('train_tiny')

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)

```



```
-----  
FileURLRetrievalError                                Traceback (most recent call last)  
/usr/local/lib/python3.10/dist-packages/gdown/download.py in download(url, output, quiet, proxy, speed, use_cookies, verify, id,  
fuzzy, resume, format, user_agent, log_messages)  
    266         try:  
--> 267             url = get_url_from_gdrive_confirmation(res.text)  
    268         except FileURLRetrievalError as e:
```

3 frames

FileURLRetrievalError: Cannot retrieve the public link of the file. You may need to change the permission to 'Anyone with the link', or have had many accesses. Check FAQ in <https://github.com/wkentaro/gdown?tab=readme-ov-file#faq>.

During handling of the above exception, another exception occurred:

```
FileURLRetrievalError                                Traceback (most recent call last)  
/usr/local/lib/python3.10/dist-packages/gdown/download.py in download(url, output, quiet, proxy, speed, use_cookies, verify, id,  
fuzzy, resume, format, user_agent, log_messages)  
    276         url_origin,  
    277         )  
--> 278         raise FileURLRetrievalError(message)  
    279  
    280     filename_from_url = None
```

FileURLRetrievalError: Failed to retrieve file url:

Cannot retrieve the public link of the file. You may need to change the permission to 'Anyone with the link', or have had many accesses. Check FAQ in <https://github.com/wkentaro/gdown?tab=readme-ov-file#faq>.

You may still be able to access the file from the browser:

<https://drive.google.com/uc?export=download&confirm=pbef&id=1I-270uXld40whZQ0ltp817Kn3J0Xgbui>

but Gdown can't. Please check connections and permissions.

✓ Класс Metrics

Реализует метрики точности, используемые для оценивания модели:

1. точность,
2. сбалансированную точность.

```
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.

Вам необходимо реализовать методы `save`, `load` для сохранения и загрузки модели. Особенно актуально это будет во время тестирования на дополнительных наборах данных.

Пожалуйста, убедитесь, что сохранение и загрузка модели работает корректно. Для этого обучите модель, протестируйте, сохраните ее в файл, перезапустите среду выполнения, загрузите обученную модель из файла, вновь протестируйте ее на тестовой выборке и убедитесь в том, что получаемые метрики совпадают с полученными для тестовой выборки ранее.

Также, Вы можете реализовать дополнительные функции, такие как:

1. валидацию модели на части обучающей выборки;
2. использование кроссвалидации;
3. автоматическое сохранение модели при обучении;

4. загрузку модели с какой-то конкретной итерации обучения (если используется итеративное обучение);
5. вывод различных показателей в процессе обучения (например, значение функции потерь на каждой эпохе);
6. построение графиков, визуализирующих процесс обучения (например, график зависимости функции потерь от номера эпохи обучения);
7. автоматическое тестирование на тестовом наборе/наборах данных после каждой эпохи обучения (при использовании итеративного обучения);
8. автоматический выбор гиперпараметров модели во время обучения;
9. сохранение и визуализацию результатов тестирования;
10. Использование аугментации и других способов синтетического расширения набора данных (дополнительным плюсом будет обоснование необходимости и обоснование выбора конкретных типов аугментации)
11. и т.д.

Полный список опций и дополнений приведен в презентации с описанием задания.

При реализации дополнительных функций допускается добавление параметров в существующие методы и добавление новых методов в класс модели.

```
class Model:

    def __init__(self):
        self.img_height = 224
        self.img_width = 224
        self.batch_size = 32
        self.num_classes = 9
        self.model = self.build_model()
        self.d_train = None
        self.d_test = None

    def build_model(self):
        base_model = EfficientNetB0(weights='imagenet', include_top=False, input_shape=(self.img_height, self.img_width, 3))
        base_model.trainable = True
        for layer in base_model.layers[:-5]:
            layer.trainable = False

        model = models.Sequential()
        model.add(base_model)
        model.add(layers.GlobalAveragePooling2D())
        model.add(layers.Dense(32, activation='relu'))
        model.add(layers.Dropout(0.25))
        model.add(layers.Dense(self.num_classes, activation='softmax'))

        model.compile(optimizer='adam',
                      loss='categorical_crossentropy',
                      metrics=['accuracy'])

        return model

    def save(self, name: str):
        self.model.save(f'/content/drive/MyDrive/{name}.h5')
        print(f'Model saved to /content/drive/MyDrive/{name}.h5')

    def load(self, name: str):
        self.model = tf.keras.models.load_model(f'/content/drive/MyDrive/{name}.h5')
        self.model.compile(optimizer='adam',
                          loss='categorical_crossentropy',
                          metrics=['accuracy'])
        print(f'Model loaded from /content/drive/MyDrive/{name}.h5')

    def train(self, dataset: Dataset):
        self.d_train = dataset
        X_train, X_val, y_train, y_val = train_test_split(dataset.images, dataset.labels, test_size=0.2, random_state=42)#доп 1 Валидац
        model_prep = models.Sequential([#доп 11 аугментация, типы выбраны опытным путем
            layers.Resizing(self.img_height, self.img_width),
            layers.RandomFlip(mode="horizontal"),
            layers.RandomRotation(0.2),
            layers.RandomZoom(height_factor=0.1, width_factor=0.1),
            layers.RandomContrast(factor=0.1)
        ])

    def train_generator():
        while True:
            indices = np.random.choice(len(X_train), self.batch_size)
            images = X_train[indices]
            labels = y_train[indices]
            images = tf.image.resize(images, [self.img_height, self.img_width])
            yield model_prep(images), tf.keras.utils.to_categorical(labels, num_classes=9)
```

```

def validation_generator():
    while True:
        indices = np.random.choice(len(X_val), self.batch_size)
        images = X_val[indices]
        labels = y_val[indices]
        images = tf.image.resize(images, [self.img_height, self.img_width])
        yield images, tf.keras.utils.to_categorical(labels, num_classes=9)
log_dir = os.path.join("logs", "fit", "model")

tensorboard_callback = TensorBoard(log_dir=log_dir, histogram_freq=1)
reduce_lr = ReduceLROnPlateau(monitor='val_accuracy', factor=0.5, patience=5, min_lr=1e-6, verbose=1)
early_stopping = EarlyStopping(monitor='val_accuracy', patience=10, restore_best_weights=True, verbose=1)# доп 4 Загрузка модели

print(f'training started')
self.model.fit(train_generator(),
               steps_per_epoch=len(X_train) // self.batch_size,
               validation_data=validation_generator(),
               validation_steps=len(X_val) // self.batch_size,
               epochs=90,
               callbacks=[tensorboard_callback, reduce_lr, early_stopping])#доп 5 Вывод различных показателей в процессе обучения
print(f'training done')
# возможно так не считается, доп 6 Построение графиков, визуализирующих процесс обучения
%load_ext tensorboard
%tensorboard --logdir {log_dir}

def test_on_dataset(self, dataset: Dataset, limit=None):
    predictions = []
    true_labels = []
    n = dataset.n_files if not limit else int(dataset.n_files * limit)
    true_labels = dataset.labels[:n]
    for img in tqdm(dataset.images_seq(n), total=n):
        predictions.append(self.test_on_image(img))
    cm = confusion_matrix(true_labels, predictions)

    plt.figure(figsize=(10, 8))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=range(self.num_classes), yticklabels=range(self.num_classes))
    plt.xlabel('Predicted labels')
    plt.ylabel('True labels')
    plt.title('Confusion Matrix')
    plt.show()

    return predictions

def test_on_image(self, img: np.ndarray):
    img = tf.image.resize(img, [self.img_height, self.img_width])
    img = np.expand_dims(img, axis=0)
    prediction = np.argmax(self.model.predict(img, verbose=0), axis=-1)[0]
    return prediction

```

Классификация изображений

Используя введенные выше классы можем перейти уже непосредственно к обучению модели классификации изображений. Пример общего пайплайна решения задачи приведен ниже. Вы можете его расширять и улучшать. В данном примере используются наборы данных 'train_small' и 'test_small'.

```

d_train = Dataset('train')
d_test = Dataset('test')

```



```

Downloading...
From: https://drive.google.com/uc?export=download&confirm=pbef&id=1jmFzZRPv7CzmUUMziRQo4VTfbYJD8DA8
To: /content/train.npz
100%|██████████| 2.10G/2.10G [00:23<00:00, 88.3MB/s]
Loading dataset train from npz.
Done. Dataset train consists of 18000 images.
Downloading...
From: https://drive.google.com/uc?export=download&confirm=pbef&id=1V6jaw1sFnow64Z9QMi6upcxtfLJ9KuUA
To: /content/test.npz
100%|██████████| 525M/525M [00:06<00:00, 79.4MB/s]
Loading dataset test from npz.
Done. Dataset test consists of 4500 images.

```

```

model = Model()
if not EVALUATE_ONLY:
    model.train(d_train)
    model.save('best')
else:
    #todo: your link goes here

```

```
model.load('best')
```

WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until y
Model loaded from /content/drive/MyDrive/best.h5

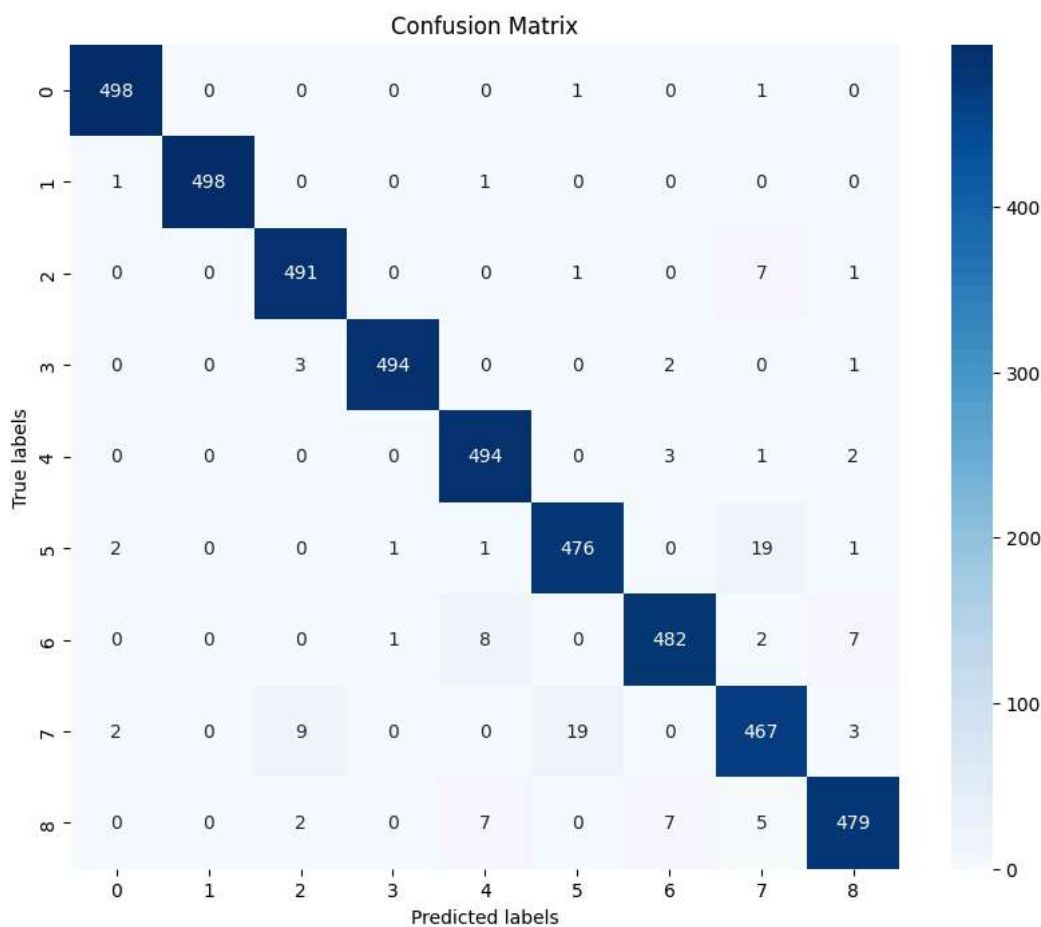
Пример тестирования модели на части набора данных:

```
# 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')
```

Пример тестирования модели на полном наборе данных:

```
# 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')
```

100% 4500/4500 [05:51<00:00, 14.07it/s]



```
metrics for test:
  accuracy 0.9731:
  balanced accuracy 0.9731:
```

Результат работы пайплайна обучения и тестирования выше тоже будет оцениваться. Поэтому не забудьте присылать на проверку ноутбук с выполненными ячейками кода с демонстрациями метрик обучения, графиками и т.п. В этом пайплайне Вам необходимо продемонстрировать работу всех реализованных дополнений, улучшений и т.п.

Настоятельно рекомендуется после получения пайплайна с полными результатами обучения экспортировать ноутбук в pdf (файл -> печать) и прислать этот pdf вместе с самим ноутбуком.

✓ Тестирование модели на других наборах данных

Ваша модель должна поддерживать тестирование на других наборах данных. Для удобства, Вам предоставляется набор данных `test_tiny`, который представляет собой малую часть (2% изображений) набора `test`. Ниже приведен фрагмент кода, который будет осуществлять тестирование для оценивания Вашей модели на дополнительных тестовых наборах данных.

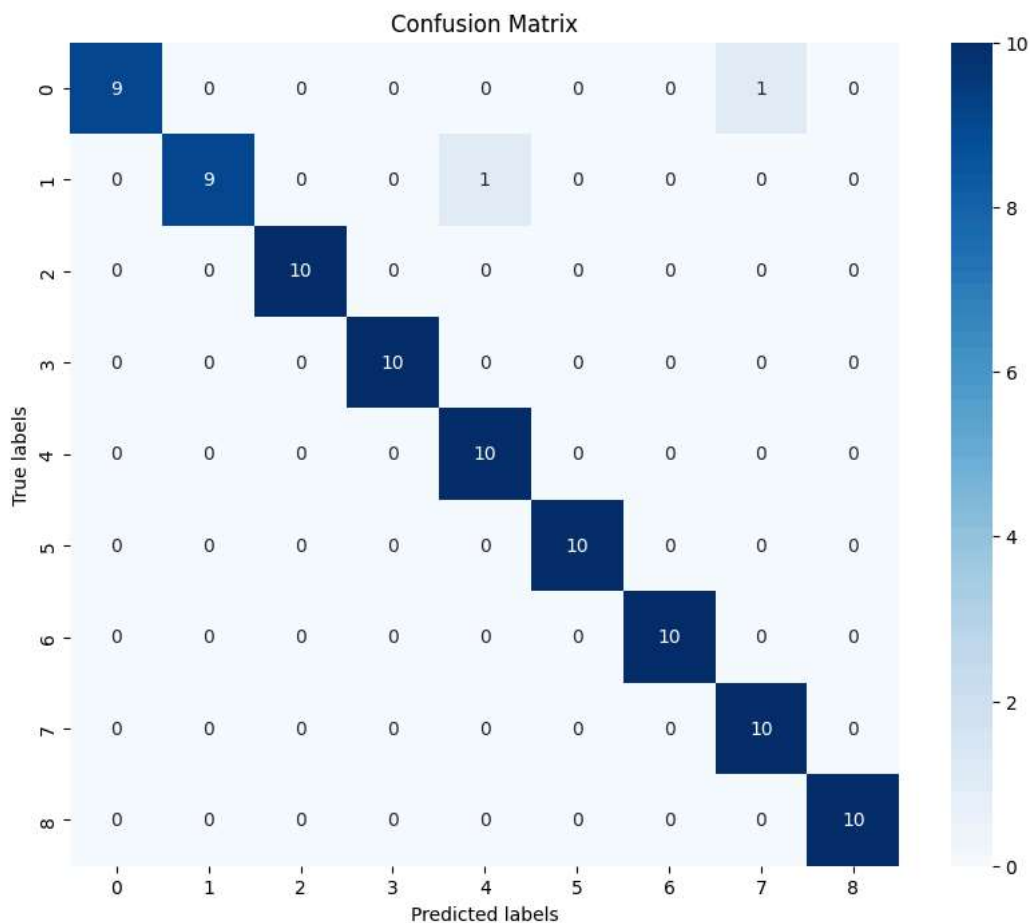
Прежде чем отсылать задание на проверку, убедитесь в работоспособности фрагмента кода ниже.

```

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

```

⚠ WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until)
Model loaded from /content/drive/MyDrive/best.h5
Downloading...
From: <https://drive.google.com/uc?export=download&confirm=pbef&id=1viiB0s041CNsAK4itvX8PnYthJ-MDnQc>
To: /content/test_tiny.npz
100%|██████████| 10.6M/10.6M [00:00<00:00, 47.1MB/s]
Loading dataset test_tiny from npz.
Done. Dataset test_tiny consists of 90 images.
100% 90/90 [00:07<00:00, 9.75it/s]



metrics for test-tiny:
accuracy 0.9778:
balanced accuracy 0.9778:

Отмонтировать Google Drive.

```
drive.flush_and_unmount()
```

✓ Дополнительные "полезности"

Ниже приведены примеры использования различных функций и библиотек, которые могут быть полезны при выполнении данного практического задания.

✓ Измерение времени работы кода

Измерять время работы какой-либо функции можно легко и непринужденно при помощи функции `timeit` из соответствующего модуля:

```

import timeit

def factorial(n):
    res = 1
    for i in range(1, n + 1):

```

```

        res *= i
    return res

def f():
    return factorial(n=1000)

n_runs = 128
print(f'Function f is calculated {n_runs} times in {timeit.timeit(f, number=n_runs)}s.')
```

✓ Scikit-learn

Для использования "классических" алгоритмов машинного обучения рекомендуется использовать библиотеку scikit-learn (<https://scikit-learn.org/stable/>). Пример классификации изображений цифр из набора данных MNIST при помощи классификатора SVM:

```

# Standard scientific Python imports
import matplotlib.pyplot as plt

# Import datasets, classifiers and performance metrics
from sklearn import datasets, svm, metrics
from sklearn.model_selection import train_test_split

# The digits dataset
digits = datasets.load_digits()

# The data that we are interested in is made of 8x8 images of digits, let's
# have a look at the first 4 images, stored in the `images` attribute of the
# dataset. If we were working from image files, we could load them using
# matplotlib.pyplot.imread. Note that each image must have the same size. For these
# images, we know which digit they represent: it is given in the `target` of
# the dataset.
_, axes = plt.subplots(2, 4)
images_and_labels = list(zip(digits.images, digits.target))
for ax, (image, label) in zip(axes[0, :], images_and_labels[:4]):
    ax.set_axis_off()
    ax.imshow(image, cmap=plt.cm.gray_r, interpolation='nearest')
    ax.set_title('Training: %i' % label)

# To apply a classifier on this data, we need to flatten the image, to
# turn the data in a (samples, feature) matrix:
n_samples = len(digits.images)
data = digits.images.reshape((n_samples, -1))

# Create a classifier: a support vector classifier
classifier = svm.SVC(gamma=0.001)

# Split data into train and test subsets
X_train, X_test, y_train, y_test = train_test_split(
    data, digits.target, test_size=0.5, shuffle=False)

# We learn the digits on the first half of the digits
classifier.fit(X_train, y_train)

# Now predict the value of the digit on the second half:
predicted = classifier.predict(X_test)

images_and_predictions = list(zip(digits.images[n_samples // 2:], predicted))
for ax, (image, prediction) in zip(axes[1, :], images_and_predictions[:4]):
    ax.set_axis_off()
    ax.imshow(image, cmap=plt.cm.gray_r, interpolation='nearest')
    ax.set_title('Prediction: %i' % prediction)

print("Classification report for classifier %s:\n%s\n"
      % (classifier, metrics.classification_report(y_test, predicted)))
disp = metrics.plot_confusion_matrix(classifier, X_test, y_test)
disp.figure_.suptitle("Confusion Matrix")
print("Confusion matrix:\n%s" % disp.confusion_matrix)

plt.show()
```

✓ Scikit-image

Реализовывать различные операции для работы с изображениями можно как самостоятельно, работая с массивами numpy, так и используя специализированные библиотеки, например, scikit-image (<https://scikit-image.org/>). Ниже приведен пример использования Canny edge detector.


```

import numpy as np
import matplotlib.pyplot as plt
from scipy import ndimage as ndi

from skimage import feature

# Generate noisy image of a square
im = np.zeros((128, 128))
im[32:-32, 32:-32] = 1

im = ndi.rotate(im, 15, mode='constant')
im = ndi.gaussian_filter(im, 4)
im += 0.2 * np.random.random(im.shape)

# Compute the Canny filter for two values of sigma
edges1 = feature.canny(im)
edges2 = feature.canny(im, sigma=3)

# display results
fig, (ax1, ax2, ax3) = plt.subplots(nrows=1, ncols=3, figsize=(8, 3),
                                    sharex=True, sharey=True)

ax1.imshow(im, cmap=plt.cm.gray)
ax1.axis('off')
ax1.set_title('noisy image', fontsize=20)

ax2.imshow(edges1, cmap=plt.cm.gray)
ax2.axis('off')
ax2.set_title(r'Canny filter,  $\sigma=1$ ', fontsize=20)

ax3.imshow(edges2, cmap=plt.cm.gray)
ax3.axis('off')
ax3.set_title(r'Canny filter,  $\sigma=3$ ', fontsize=20)

fig.tight_layout()

plt.show()

```

✓ Tensorflow 2

Для создания и обучения нейросетевых моделей можно использовать фреймворк глубокого обучения Tensorflow 2. Ниже приведен пример простейшей нейронной сети, использующейся для классификации изображений из набора данных MNIST.

```

# Install TensorFlow

import tensorflow as tf

mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)

model.evaluate(x_test, y_test, verbose=2)

```

Для эффективной работы с моделями глубокого обучения убедитесь в том, что в текущей среде Google Colab используется аппаратный ускоритель GPU или TPU. Для смены среды выберите "среда выполнения" -> "сменить среду выполнения".

Большое количество tutorиалов и примеров с кодом на Tensorflow 2 можно найти на официальном сайте <https://www.tensorflow.org/tutorials?hl=ru>.

Также, Вам может понадобиться написать собственный генератор данных для Tensorflow 2. Скорее всего он будет достаточно простым, и его легко можно будет реализовать, используя официальную документацию TensorFlow 2. Но, на всякий случай (если не

удлось сразу разобраться или хочется вникнуть в тему более глубоко), можете посмотреть следующий отличный tutorial: <https://stanford.edu/~shervine/blog/keras-how-to-generate-data-on-the-fly>.

Numba

В некоторых ситуациях, при ручных реализациях графовых алгоритмов, выполнение многократных вложенных циклов for в python можно существенно ускорить, используя JIT-компилятор Numba (<https://numba.pydata.org/>). Примеры использования Numba в Google Colab можно найти тут:

1. https://colab.research.google.com/github/cbernet/maldives/blob/master/numba/numba_cuda.ipynb
2. https://colab.research.google.com/github/evaneschneider/parallel-programming/blob/master/COMPASS_gpu_intro.ipynb

Пожалуйста, если Вы решили использовать Numba для решения этого практического задания, еще раз подумайте, нужно ли это Вам, и есть ли возможность реализовать требуемую функциональность иным способом. Используйте Numba только при реальной необходимости.

✎ Работа с zip архивами в Google Drive

Запаковка и распаковка zip архивов может пригодиться при сохранении и загрузки Вашей модели. Ниже приведен фрагмент кода, иллюстрирующий помещение нескольких файлов в zip архив с последующим чтением файлов из него. Все действия с директориями, файлами и архивами должны осуществляться с примонтированным Google Drive.

Создадим 2 изображения, поместим их в директорию tmp внутри PROJECT_DIR, запакуем директорию tmp в архив tmp.zip.

```
PROJECT_DIR = "/dev/prak_nn_1/"
arr1 = np.random.rand(100, 100, 3) * 255
arr2 = np.random.rand(100, 100, 3) * 255

img1 = Image.fromarray(arr1.astype('uint8'))
img2 = Image.fromarray(arr2.astype('uint8'))

p = "/content/drive/MyDrive/" + PROJECT_DIR

if not (Path(p) / 'tmp').exists():
    (Path(p) / 'tmp').mkdir()

img1.save(str(Path(p) / 'tmp' / 'img1.png'))
img2.save(str(Path(p) / 'tmp' / 'img2.png'))

%cd $p
!zip -r "tmp.zip" "tmp"
```

Распакуем архив tmp.zip в директорию tmp2 в PROJECT_DIR. Теперь внутри директории tmp2 содержится директория tmp, внутри которой находятся 2 изображения.

```
p = "/content/drive/MyDrive/" + PROJECT_DIR
%cd $p
```