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 (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 gdown) (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
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (2023.7.2
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (1.7
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:

```
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': '1XtQzVQ5XbrfxpLHJuL0XBGJ5U7CS-cLi',
    'train_small': '1qd45xXfDwdZjktLFwQb-et-mAaFeCzOR',
    'train_tiny': '1I-2ZOuXLd4QwhZQQltp817Kn3J0Xgbui',
    'test': '1RfPou3pFKpuHDJZ-D9XDFzgvwpUBFlDr',
    'test_small': '1wbRsog0n7uGlHIPGLhyN-PMeT2kdQ2lI',
    'test_tiny': '1Ebi6g7tXGCOxJG6_M3OCXaBWH1xYC_AS'

}
```

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 torch.utils.data import DataLoader
import torch.utils.data as utils
from torchvision import transforms
```

```
!pip install efficientnet_pytorch
```

```
Collecting efficientnet_pytorch
  Downloading efficientnet_pytorch-0.7.1.tar.gz (21 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packages (from efficientnet_pytorch) (2.1.0+cu118)
```

```
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch->efficientnet_pytorch) (3.13.1)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.10/dist-packages (from torch->efficientnet_pytorch) (4.5
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch->efficientnet_pytorch) (1.12)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch->efficientnet_pytorch) (3.2.1)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch->efficientnet_pytorch) (3.1.2)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from torch->efficientnet_pytorch) (2023.6.0)
Requirement already satisfied: triton==2.1.0 in /usr/local/lib/python3.10/dist-packages (from torch->efficientnet_pytorch) (2.1.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch->efficientnet_pytorch
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist-packages (from sympy->torch->efficientnet_pytorch) (1
Building wheels for collected packages: efficientnet_pytorch
  Building wheel for efficientnet_pytorch (setup.py) ... done
  Created wheel for efficientnet_pytorch: filename=efficientnet_pytorch-0.7.1-py3-none-any.whl size=16428 sha256=187dc34c266b9ae048e
  Stored in directory: /root/.cache/pip/wheels/03/3f/e9/911b1bc46869644912bda90a56bcf7b960f20b5187feea3baf
Successfully built efficientnet_pytorch
Installing collected packages: efficientnet_pytorch
Successfully installed efficientnet_pytorch-0.7.1
```

```
import torch
import torchvision
from tqdm.auto import tqdm
from efficientnet_pytorch import EfficientNet
import matplotlib.pyplot as plt
%matplotlib inline
```

Класс Dataset

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

```
data_transform = transforms.Compose(
    [transforms.ToPILImage(), transforms.ToTensor()]
)
```

```
class Dataset(torch.utils.data.Dataset):

    def __init__(self, name, transform=data_transform):
        super().__init__()
        self.transform = transform
        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 __len__(self):
        return self.n_files

    def __getitem__(self, index: int):
        return self.image_with_label(index)

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

Пример использвания 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.

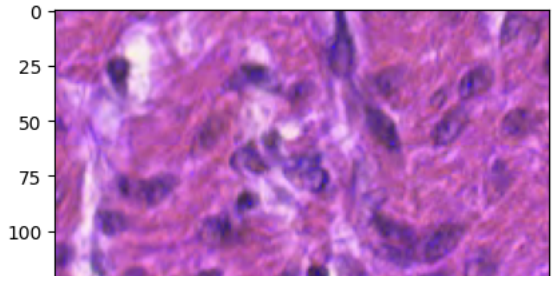
```
d_train_tiny = Dataset('train_tiny', transform=data_transform)

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)
plt.imshow(img.permute(1, 2, 0))
```

```
Downloading...
From: https://drive.google.com/uc?export=download&confirm=pbef&id=1I-2ZOuXLd4QwhZQQlt
To: /content/train_tiny.npz
100%|██████████| 105M/105M [00:03<00:00, 29.6MB/s]
Loading dataset train_tiny from npz.
Done. Dataset train_tiny consists of 900 images.

Got numpy array of shape torch.Size([3, 224, 224]), and label with code 5.
Label code corresponds to MUS class.
<matplotlib.image.AxesImage at 0x78c4f2affcd0>
```



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 заргрузки 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 выбрки 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.

```
#LBL2
train_transform = transforms.Compose([
    transforms.ToPILImage(),
    transforms.ToTensor(),
    transforms.RandomHorizontalFlip(),
    transforms.RandomRotation(20)
    #transforms.Normalize((0.485, 0.456, 0.406), (0.229, 0.224, 0.225))
])

test_transform = transforms.Compose([
    transforms.ToPILImage(),
    transforms.ToTensor(),
    #transforms.Normalize((0.485, 0.456, 0.406), (0.229, 0.224, 0.225))
])



d_train = Dataset('train', transform=train_transform)
```

```
Downloading...
From: https://drive.google.com/uc?export=download&confirm=pbef&id=1XtQzVQ5XbrfxpLHJuL0XBGJ5U7CS-cLi
To: /content/train.npz
100%|██████████| 2.10G/2.10G [00:25<00:00, 82.5MB/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=1wbRsog0n7uGlHIPGLhyN-PMeT2kdQ2lI
To: /content/test_small.npz
100%|██████████| 211M/211M [00:04<00:00, 51.1MB/s]
Loading dataset test_small from npz.
Done. Dataset test_small consists of 1800 images.
```

```
d_test = Dataset('test', transform=test_transform)
```

```
Downloading...
From: https://drive.google.com/uc?export=download&confirm=pbef&id=1RfPou3pFKpuHDJZ-D9XDFzgvwpUBFlDr
To: /content/test.npz
100%|██████████| 525M/525M [00:07<00:00, 68.7MB/s]
Loading dataset test from npz.
Done. Dataset test consists of 4500 images.
```

```
dl_train = DataLoader(dataset = d_train, shuffle=True, num_workers=2, batch_size = 64)


class Model:

    def __init__(self):
        self.path_to_best = '1s5m8WKzT5NaGLDpKPSosUYi33BxKoe2N'
        self.n_classes = 9
        self.model = torchvision.models.efficientnet_b1(weights='DEFAULT')
        self.model.classifier = torch.nn.Sequential(
            torch.nn.Dropout(0.2),
            torch.nn.Linear(1280, 1000),
            torch.nn.BatchNorm1d(1000),
            torch.nn.SiLU(),
            torch.nn.Linear(1000, self.n_classes)
        )

    def save(self, name: str):
        torch.save(self.model.state_dict(), f'/content/drive/MyDrive/med_images/{name}.pth')

    def set_path_to_best(self, path: str):
        self.path_to_best = path

    def load(self, name: str):
        name_to_id_dict = {
            'best': self.path_to_best,
        }
        output = f'{name}.pth'
        gdown.download(f'https://drive.google.com/uc?id={name_to_id_dict[name]}', output, quiet=False)
        self.model.load_state_dict(torch.load(output))

    def train(self,
              device,
              n_epochs,
              dl_train
              ):
```

```
        ).
        print(f'training started')

        optimizer=torch.optim.Adam(self.model.parameters())
        self.model.to(device)
        train_losses, test_accuracies = [], []

        for epoch in range(n_epochs):
            print("epoch ", epoch)
            correct_epoch = 0
            items_epoch = 0
            loss_epoch = []
            self.model.train()
            for images, labels in tqdm(dl_train):
                images, labels = images.to(device), labels.type(torch.LongTensor).to(device)
                optimizer.zero_grad()
                y_pred = self.model(images)
                loss = torch.nn.functional.cross_entropy(y_pred, labels)
                loss.backward()
                loss_epoch.append(loss.flatten()[0].item())
                optimizer.step()
                correct_epoch += (torch.argmax(y_pred, dim=1) == labels).sum().item()
                items_epoch += labels.size(0)
            train_losses.append(sum(loss_epoch)/len(loss_epoch))

            #LBL3
            self.model.eval()
            num_correct = 0
            num_all = 0
            dl_test_small = DataLoader(dataset = Dataset('test_small'), shuffle=False, num_workers=2, batch_size = 64)
            for X, y in tqdm(dl_test_small):
                X = X.to(device)
                y = y.to(device)
                num_all += len(y)
                with torch.no_grad():
                    logits = self.model(X)
                    y_pred = logits.argmax(-1)
                    num_correct += (y_pred == y).sum()
            test_accuracies.append(num_correct / num_all)
            #LBL1
            print(
            'Loss (Train): {0:.3f}. Accuracy, % (Test_small): {1:.2f}\n'.format(
                train_losses[-1], test_accuracies[-1]
            )
        )
        print(f'training done')
        return train_losses, test_accuracies

    def test_on_dataset(self, dataset: Dataset, limit=None):
        predictions = []
        dl_test = DataLoader(dataset = dataset, shuffle=False, num_workers=2, batch_size = 64)
        device = torch.device('cpu')
        if torch.cuda.is_available():
            device = torch.device('cuda', 0)
        self.model = self.model.to(device)
        self.model.eval()
        with torch.no_grad():
            for images, labels in tqdm(dl_test):
                images = images.to(device)
                labels = labels.type(torch.LongTensor)
                labels = labels.to(device)
                y_pred = self.model(images)
                [predictions.append(elem) for elem in torch.argmax(y_pred, dim=1).cpu().tolist()]
        return predictions
```

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' (изменила на train) и 'test_small'.

```
device = torch.device('cpu')
if torch.cuda.is_available():
    device = torch.device('cuda', 0)

print(type(device), device)
```

```
<class 'torch.device'> cuda:0
```

```
model = Model()
model.model = model.model.to(device)
```

```
train_losses, test_accs = model.train(dl_train=dl_train, device=device, n_epochs=1)
```

```
training started
epoch  0
100%                                                282/282 [03:08<00:00, 2.58it/s]
Downloading...
From: https://drive.google.com/uc?export=download&confirm=pbef&id=1wbRsog0n7uGlHIPGLhyN-PMeT2kdQ2lI
To: /content/test_small.npz
100%|██████████| 211M/211M [00:00<00:00, 282MB/s]
Loading dataset test_small from npz.
Done. Dataset test_small consists of 1800 images.
100%                                                29/29 [00:08<00:00, 5.63it/s]
Loss (Train): 0.169. Accuracy, % (Test_small): 0.98

training done
```

```
model.save('best')
```

```
model.set_path_to_best('1s5m8WKzT5NaGLDpKPSosUYi33BxKoe2N')
```

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

```
Downloading...
From (uriginal): https://drive.google.com/uc?id=1s5m8WKzT5NaGLDpKPSosUYi33BxKoe2N
From (redirected): https://drive.google.com/uc?id=1s5m8WKzT5NaGLDpKPSosUYi33BxKoe2N&confirm=t&uuid=a3a1a8c0-b717-4880-831c-723aa4c5
To: /content/best.pth
100%|██████████| 31.6M/31.6M [00:00<00:00, 191MB/s]
```

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)
# pred_1
```

```
100%                                                29/29 [00:07<00:00, 5.77it/s]
```

```
# Metrics.print_all(d_test.labels[:len(pred_1)], pred_1, '10% of test')
```

```
metrics for 10% of test:
         accuracy 0.9789:
         balanced accuracy 0.9789:
```

Пример тестирования модели на полном наборе данных:

```
model.model = model.model.to(device)
```

```
# 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%                                                71/71 [00:17<00:00, 6.17it/s]
metrics for test:
         accuracy 0.9796:
         balanced accuracy 0.9796:
```

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 = final_model.test_on_dataset(d_test_tiny)
Metrics.print_all(d_test_tiny.labels, pred, 'test-tiny')
```

```
Downloading...
From (uriginal): https://drive.google.com/uc?id=1s5m8WKzT5NaGLDpKPSosUYi33BxKoe2N
From (redirected): https://drive.google.com/uc?id=1s5m8WKzT5NaGLDpKPSosUYi33BxKoe2N&c
To: /content/best.pth
100%|██████████| 31.6M/31.6M [00:00<00:00, 137MB/s]
Downloading...
From: https://drive.google.com/uc?export=download&confirm=pbef&id=1Ebi6g7tXGCOxJG6_M3
To: /content/test_tiny.npz
100%|██████████| 10.6M/10.6M [00:00<00:00, 249MB/s]Loading dataset test_tiny from npz
Done. Dataset test_tiny consists of 90 images.

100%                                                2/2 [00:01<00:00, 1.12it/s]
metrics for test-tiny:
        accuracy 0.9556:
```

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 caluclated {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 нейроной 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 туториалов 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 туториал: 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