# Практическое задание №1

Установка необходимых пакетов:

```
#install python 3.9
!sudo apt-get update -y
!sudo apt-get install python3.9

#change alternatives
!sudo update-alternatives --install /usr/bin/python3 python3 /usr/bin/python3.7 1
!sudo update-alternatives --install /usr/bin/python3 python3 /usr/bin/python3.9 2

#check python version
!python --version
```

```
Hit:15 http://archive.ubuntu.com/ubuntu bionic InRelease
Get:16 http://archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:17 http://security.ubuntu.com/ubuntu bionic-security/restricted amd64 Packages [1,294 kB]
Get:18 http://archive.ubuntu.com/ubuntu bionic-backports InRelease [83.3 kB]
Get:19 http://archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages [2,342 kB]
Get:20 http://archive.ubuntu.com/ubuntu bionic-updates/restricted amd64 Packages [1,334 kB]
Get:21 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [3,519 kB]
Fetched 14.5 MB in 2min 27s (98.4 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree
Reading state information... Done

The following package was automatically installed and is no longer required:
  libnvidia-common-460
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  libpython3.9-minimal libpython3.9-stdlib python3.9-minimal
Suggested packages:
  python3.9-venv binfmt-support
The following NEW packages will be installed:
  libpython3.9-minimal libpython3.9-stdlib python3.9 python3.9-minimal
0 upgraded, 4 newly installed, 0 to remove and 14 not upgraded.
Need to get 4,917 kB of archives.
After this operation, 19.1 MB of additional disk space will be used.
Get:1 http://ppa.launchpad.net/deadsnakes/ppa/ubuntu bionic/main amd64 libpython3.9-minimal amd64 3.9.15-1+bionic1 [805 kB]
Get:2 http://ppa.launchpad.net/deadsnakes/ppa/ubuntu bionic/main amd64 python3.9-minimal amd64 3.9.15-1+bionic1 [1,938 kB]
```

```
Get:3 http://ppa.launchpad.net/deadsnakes/ppa/ubuntu bionic/main amd64 libpython3.9-stdlib amd64 3.9.15-1+bionic1 [1,680 kB]
Get:4 http://ppa.launchpad.net/deadsnakes/ppa/ubuntu bionic/main amd64 python3.9 amd64 3.9.15-1+bionic1 [494 kB]
Fetched 4,917 kB in 8s (606 kB/s)
debconf: unable to initialize frontend: Dialog
debconf: (No usable dialog-like program is installed, so the dialog based frontend cannot be used. at /usr/share/perl5/Debconf/FrontE
debconf: falling back to frontend: Readline
debconf: unable to initialize frontend: Readline
debconf: (This frontend requires a controlling tty.)
debconf: falling back to frontend: Teletype
dpkg-preconfigure: unable to re-open stdin:
Selecting previously unselected package libpython3.9-minimal:amd64.
(Reading database ... 124015 files and directories currently installed.)
Preparing to unpack .../libpython3.9-minimal_3.9.15-1+bionic1_amd64.deb ...
Unpacking libpython3.9-minimal:amd64 (3.9.15-1+bionic1) ...
Selecting previously unselected package python3.9-minimal.
Preparing to unpack .../python3.9-minimal_3.9.15-1+bionic1_amd64.deb ...
Unpacking python3.9-minimal (3.9.15-1+bionic1) ...
Selecting previously unselected package libpython3.9-stdlib:amd64.
Preparing to unpack .../libpython3.9-stdlib_3.9.15-1+bionic1_amd64.deb ...
Unpacking libpython3.9-stdlib:amd64 (3.9.15-1+bionic1) ...
Selecting previously unselected package python3.9.
Preparing to unpack .../python3.9_3.9.15-1+bionic1_amd64.deb ...
Unpacking python3.9 (3.9.15-1+bionic1) ...
Setting up libpython3.9-minimal:amd64 (3.9.15-1+bionic1) ...
Setting up libpython3.9-stdlib:amd64 (3.9.15-1+bionic1) ...
Setting up python3.9-minimal (3.9.15-1+bionic1) ...
Setting up python3.9 (3.9.15-1+bionic1) ...
Processing triggers for mime-support (3.60ubuntu1) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
update-alternatives: error: alternative path /usr/bin/python3.7 doesn't exist
Python 3.8.15
```

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': '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

import torch
from torch.utils import data
from torch import nn
import os
from collections import namedtuple
import matplotlib.pyplot as plt
```

```
DEVICE = 'cuda' if torch.cuda.is_available() else 'cpu'
DEVICE
```

```
'cuda'
```

---

## Класс Dataset

Предназначен для работы с наборами данных, обеспечивает чтение изображений и соответствующих меток, а также формирование

```
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.')
        print('before')
        np_obj = np.load(f'{name}.npz')
        print('after')
        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
          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]

class ImageDataset(torch.utils.data.Dataset):
    def __init__(self, data: Dataset):
        self.images = data.images
        self.labels = data.labels
        self.n_files = data.n_files

    def __len__(self):
      return self.n_files

    def __getitem__(self, idx):
      image = self.image(idx)
      image = torch.tensor(image.transpose(2, 0, 1), dtype=torch.float32) / 255
      return image.to(DEVICE), self.labels[idx]

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

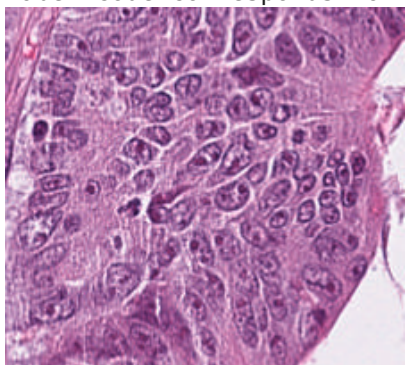
```
d_train_tiny = Dataset('train_tiny')

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)
```

```
Downloading...
From: https://drive.google.com/uc?export=download&confirm=pbef&id=1I-2ZOuXLd4QwhZQ
To: /content/train_tiny.npz
100%|██████████| 105M/105M [00:00<00:00, 277MB/s]
Loading dataset train_tiny from npz.
before
after
Done. Dataset train_tiny consists of 900 images.

Got numpy array of shape (224, 224, 3), and label with code 8.
Label code corresponds to TUM class.
```



---

## 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.

```
class ConvNet(torch.nn.Module):
    cfg = [32, "M", 64, 128, "M", 256, 256, "M"]

    def __init__(self, n_classes=10, use_batchnorm=False, dropout_p=0.0):
        '''
        :param int n_classes: Число выходных признаков
        :param bool use_batchnorm: Использовать ли батчнорм между свёрточными слоями
        :param float dropout_p: Вероятность обнуления активации слоем Dropout
        '''
        super().__init__()

        self.n_classes = n_classes

        features = []
        in_channels = 3
        for layer in self.cfg:
            if layer == 'M':
                features.append(torch.nn.MaxPool2d(kernel_size=2))
            else:
                features.append(torch.nn.Conv2d(in_channels=in_channels, out_channels=layer, kernel_size=3, padding=1))
                if use_batchnorm:
                    features.append(torch.nn.BatchNorm2d(layer))
                in_channels = layer
            features.append(torch.nn.ReLU())


        self.features = torch.nn.Sequential(*features)
        self.avgpool = torch.nn.AdaptiveAvgPool2d((2, 2))
        self.classifier = torch.nn.Sequential(
            torch.nn.Linear(256 * 2 * 2, 256),
            torch.nn.ReLU(),
            torch.nn.Dropout(p=dropout_p),
            torch.nn.Linear(256, 128),
            torch.nn.ReLU(),
            torch.nn.Dropout(p=dropout_p),
            torch.nn.Linear(128, self.n_classes)
        )
```

```
    def forward(self, x: torch.Tensor) -> torch.Tensor:
        x = self.features(x)
        x = self.avgpool(x)
        x = x.view(x.size(0), -1)
        x = self.classifier(x)
        return x
```

```
class Model:
    def __init__(self, cfg):
        self.cfg = cfg
        self.device = cfg.device
        self.out_dir = os.path.join(cfg.out_dir, 'ConvNet')
        os.makedirs(self.out_dir, exist_ok=True)

        self.model = ConvNet(9, True)
        self.criterion = nn.CrossEntropyLoss()


    def save(self, path: str):
        torch.save(self.model.state_dict(), path)


    def load(self, name: str):
      # https://drive.google.com/file/d/10217jS5uJyDIiOMUX3yRjvyvH1mAPZkj/view?usp=sharing
        name_to_id_dict = '10217jS5uJyDIiOMUX3yRjvyvH1mAPZkj'
        output = f'{name}.pth'
        gdown.download(f'https://drive.google.com/uc?id={name_to_id_dict}', output, quiet=False)
        self.model.load_state_dict(torch.load(output, map_location='cpu'))
        pass


    def load_pretrained(self, path: str):
        self.model.load_state_dict(torch.load(path))


    def train(self, train_ds: Dataset, val_ds: Dataset):
        self.model.to(self.device)
        params = [p for p in self.model.parameters() if p.requires_grad]
        optimizer = torch.optim.Adam(params, self.cfg.lr)
        lr_scheduler = torch.optim.lr_scheduler.MultiStepLR(optimizer, milestones=self.cfg.milestones)

        train_dl = torch.utils.data.DataLoader(ImageDataset(train_ds), batch_size=64, shuffle=True)
```

```
        val_dl = torch.utils.data.DataLoader(ImageDataset(val_ds), batch_size=64, shuffle=False)

        weights_dir = os.path.join(self.out_dir, "weights")
        os.makedirs(weights_dir, exist_ok=True)

        losses = []
        accuracies = []

        for epoch in (pbar := tqdm(range(self.cfg.epochs), total=self.cfg.epochs, leave=False)):

            loss = self.train_epoch(optimizer, train_dl, epoch)
            lr_scheduler.step()

            if epoch % 3 == 0:
              self.save(os.path.join(weights_dir, f"ConvNet_epoch_{epoch}.pth"))

            metrics = self.evaluate(val_dl)
            losses.append(loss)
            accuracies.append(100 * metrics['accuracy'])

            pbar.set_description(
              'Loss (Train): {0:.3f}. Accuracy, % (Test): {1:.3f}. BalancedAccuracy, % (Test): {1:.3f}\n'.format(
                  loss, metrics['accuracy'], metrics['balanced_acc']
              )
            )

        self.save(os.path.join(weights_dir, f"ConvNet.pth"))
        return losses, accuracies


    def train_epoch(self, optimizer, train_dl, epoch):
        self.model.train()

        for images, targets in tqdm(train_dl):
            optimizer.zero_grad()
            loss = self.criterion(self.model(images.to(self.device)), targets.to(self.device))

            loss.backward()
            optimizer.step()

        return loss.detach()
```

```
    def evaluate(self, val_dl):
        self.model.eval()

        prediction, target = [], []
        for images, targets in tqdm(val_dl):
            outputs = self.model(images.to(self.device)).to('cpu')
            target += list(targets.numpy())
            _, predicted = torch.max(outputs, 1)
            prediction += list(predicted.detach().numpy())

        acc = Metrics.accuracy(target, prediction)
        balanced_acc = Metrics.accuracy_balanced(target, prediction)

        return {"accuracy": acc, "balanced_acc": balanced_acc}


    def test_on_dataset(self, dataset: ImageDataset, limit=None):
        self.model.eval()
        self.model.to('cpu')
        predictions = []
        n = dataset.n_files if not limit else int(dataset.n_files * limit)
        for img in tqdm(dataset.images_seq(n), total=n):
            predictions.append(self.test_on_image(img))
        return predictions


    def test_on_image(self, img: np.ndarray):
        img = torch.tensor(img.transpose(2, 0, 1), dtype=torch.float32) / 255
        _, prediction = torch.max(self.model(img.unsqueeze(0)), 1)
        return prediction.numpy()
```

---

## 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=1XtQzVQ5XbrfxpLHJuL0XBGJ5U7CS-cLi
To: /content/train.npz
100%|██████████| 2.10G/2.10G [00:49<00:00, 42.7MB/s]
Loading dataset train from npz.
before
after
Done. Dataset train consists of 18000 images.
Downloading...
From: https://drive.google.com/uc?export=download&confirm=pbef&id=1RfPou3pFKpuHDJZ-D9XDFzgvwpUBFlDr
To: /content/test.npz
100%|██████████| 525M/525M [00:04<00:00, 110MB/s]
Loading dataset test from npz.
before
after
Done. Dataset test consists of 4500 images.
```

```
cfg_dict = {
    "out_dir": Path('drive/MyDrive/cv_data/'),
    "device": DEVICE,
    "epochs": 20,
    "lr": 0.001,
    "milestones": [6, 8, 9],
}

cfg = namedtuple("Config", cfg_dict.keys())(**cfg_dict)
EVALUATE_ONLY = False
```

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

100% 282/282 [01:49<00:00, 2.96it/s]

100% 71/71 [00:10<00:00, 6.87it/s]

100% 282/282 [01:44<00:00, 2.98it/s]

100% 71/71 [00:10<00:00, 6.58it/s]

100% 282/282 [01:44<00:00, 2.96it/s]

100% 71/71 [00:10<00:00, 6.88it/s]

100% 282/282 [01:44<00:00, 2.98it/s]

100% 71/71 [00:10<00:00, 6.98it/s]

100% 282/282 [01:44<00:00, 2.98it/s]

100% 71/71 [00:10<00:00, 6.91it/s]

100% 282/282 [01:44<00:00, 2.98it/s]

100% 71/71 [00:10<00:00, 7.04it/s]

100% 282/282 [01:44<00:00, 2.97it/s]

100% 71/71 [00:10<00:00, 6.90it/s]

100% 282/282 [01:44<00:00, 2.97it/s]

100% 71/71 [00:10<00:00, 6.88it/s]

100% 282/282 [01:44<00:00, 2.97it/s]

100% 71/71 [00:10<00:00, 6.88it/s]

100% 282/282 [01:44<00:00, 2.96it/s]

100% 71/71 [00:10<00:00, 6.80it/s]

100% 282/282 [01:44<00:00, 2.96it/s]

100% 71/71 [00:10<00:00, 6.88it/s]

100% 282/282 [01:44<00:00, 2.96it/s]

100% 71/71 [00:10<00:00, 6.82it/s]

100% 282/282 [01:44<00:00, 2.96it/s]

```
100%  71/71 [00:10<00:00, 6.88it/s]
100%  282/282 [01:44<00:00, 2.96it/s]
100%  71/71 [00:10<00:00, 6.95it/s]
100%  282/282 [01:44<00:00, 2.95it/s]
100%  71/71 [00:10<00:00, 6.86it/s]
100%  282/282 [01:44<00:00, 2.96it/s]
100%  71/71 [00:10<00:00, 6.96it/s]
100%  282/282 [01:44<00:00, 2.95it/s]
100%  71/71 [00:10<00:00, 6.89it/s]
100%  282/282 [01:44<00:00, 2.97it/s]
100%  71/71 [00:10<00:00, 6.88it/s]
100%  282/282 [01:44<00:00, 2.95it/s]
```

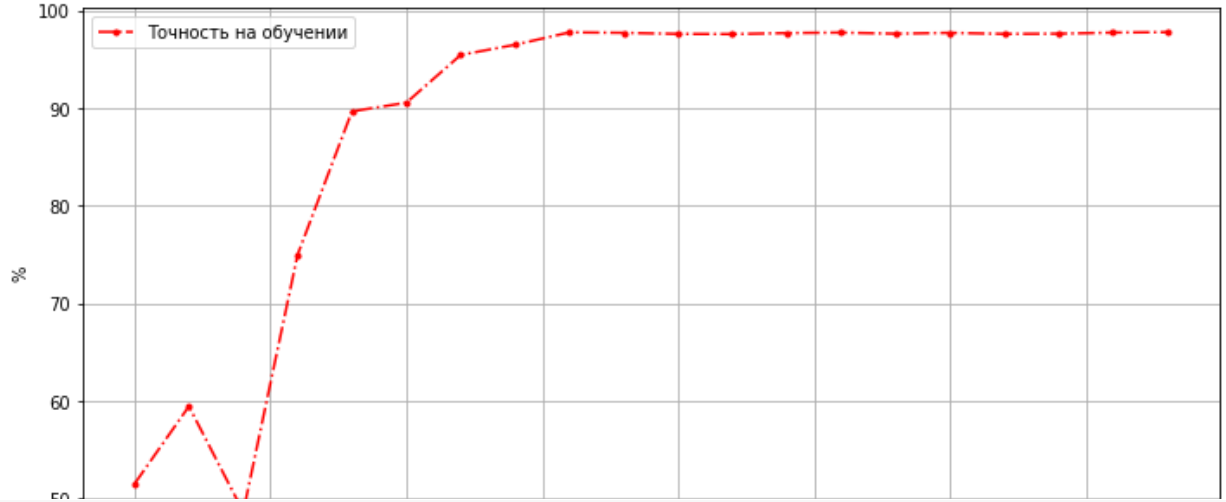
```
fig, ax = plt.subplots(1, 1, figsize=(10, 5))
ax.plot(acc, label="Точность на обучении", color='red', marker='.', linestyle='-.')


ax.set_xlabel("Номер эпохи")
ax.set_ylabel("$\%$")

ax.grid(True)
ax.legend()

fig.text(
    0.5, 0.5, '',
    fontsize=40, color='gray', alpha=0.6,
    ha='center', va='center', rotation='30'
)

fig.tight_layout()
plt.show()
```



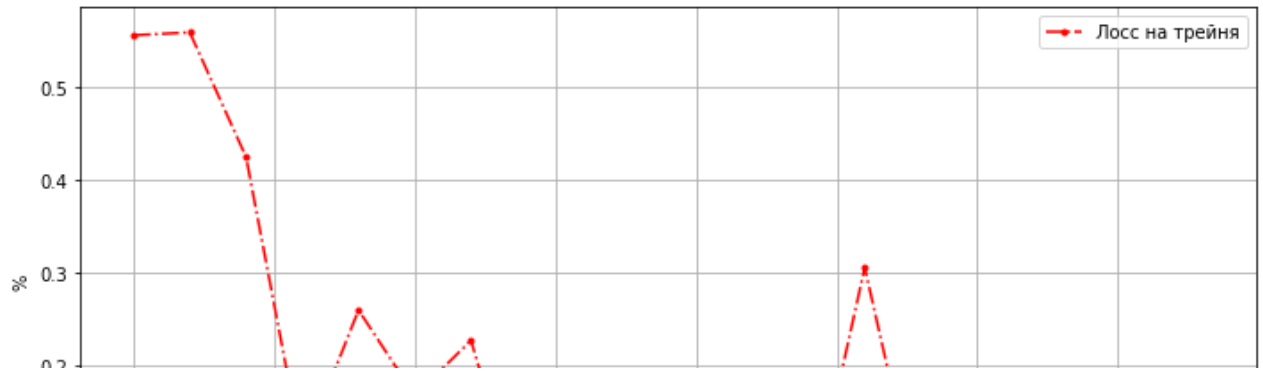

```
fig, ax = plt.subplots(1, 1, figsize=(10, 5))
ax.plot([a.item() for a in loss], label="Лосс на трейня", color='red', marker='.', linestyle='-.')


ax.set_xlabel("Номер эпохи")
ax.set_ylabel("$\%$")

ax.grid(True)
ax.legend()

fig.text(
    0.5, 0.5, '',
    fontsize=40, color='gray', alpha=0.6,
    ha='center', va='center', rotation='30'
)

fig.tight_layout()
plt.show()
```




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')
```

```
100%                                                450/450 [00:52<00:00, 8.48it/s]
metrics for 10% of test:
         accuracy 0.9956:
         balanced accuracy 0.9956:
/usr/local/lib/python3.8/dist-packages/sklearn/metrics/_classification.py:1987: UserWarning: 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:

```
# 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 [08:57<00:00, 8.55it/s]
metrics for test:
         accuracy 0.9776:
         balanced accuracy 0.9776:
```

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

## 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(cfg)
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')
```

```
Downloading...
From: https://drive.google.com/uc?id=10217jS5uJyDIiOMUX3yRjvyvH1mAPZkj
To: /content/best.pth
100%|██████████| 5.12M/5.12M [00:00<00:00, 260MB/s]
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, 214MB/s]Loading dataset test_tiny from npz.
before
after
Done. Dataset test_tiny consists of 90 images.

100%                                                  90/90 [00:10<00:00, 8.54it/s]
metrics for test-tiny:
        accuracy 0.9444:
        balanced accuracy 0.9444:
```

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.

[ ] ↳ *Скрыто 17 ячеек.*