Практическое задание №1

Файзуллов Айрат 402 группа

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-packag
Collecting gdown
  Downloading gdown-4.7.1-py3-none-any.whl (15 kB)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: requests[socks] in /usr/local/lib/python3.10/d
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/di
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dis
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/pyt
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.1
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.1
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in /usr/local/lib/pytho
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', 'T
DATASETS_LINKS = {
    'train': '13-p56cptVKjfLtLsr9i8cQ_JIcahIcMV',
    'train_small': '1FosyL3b5rnMBaglli3ZNLDC6uNccaat5',
    'train_tiny': '1SxcuL7q7zOteDDlf07rIdkgKzavzcUNf',
```

```
    'test': '1J0gdEUQliY0YQiF3k_PxEzX61KXi1nJo',
    'test_small': '1vlgiKd69h_czN8SAWxF8WiBszyqX2y4J',
    'test_tiny': '1PWhpord7PTih_QQrqGJYwlCfGnO9tRq_'
}
```

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
```

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=
#         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]
```

LB1

```
import torch
import torchvision.transforms.v2 as A
from torch.utils.data import Dataset, DataLoader
from torchvision import transforms
from torchvision.datasets import ImageFolder
import torchvision.models as models
import torch.nn as nn


import os
from urllib.request import urlretrieve
```

```
class ToTensor:
    """Convert ndarrays in sample to Tensors."""

    def __call__(self, sample):
        return torch.from_numpy(sample)


class Dataset:

  def __init__(self, name, image_transform=None, label_transform=None):
    self.name = name
    self.is_loaded = False
    url = f"https://drive.google.com/uc?export=download&confirm=pbef&id={DATAS
    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', allow_pickle=True)
    self.images = np_obj['data']
    self.labels = np_obj['labels']
    self.n_files = self.images.shape[0]
```

```
    self.is_loaded = True

    if not image_transform:
      self._images = torch.tensor(self.images, dtype=torch.float32)
    else:
      self._images = image_transform(self.images.astype(np.float32))

    if not label_transform:
      self._labels = torch.tensor(self.labels)
    else:
      self._labels = label_transform(self.labels)

    print(f'Done. Dataset {name} consists of {self.n_files} images.')

  def __len__(self,):

        length = len(self._labels)

        return length

  def __getitem__(self, idx):

      img = self._images[idx].permute(2, 1, 0)
      label = self._labels[idx]
      return img, label

```

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.

LB 2

Пример изображения

```
from matplotlib import pyplot as plt
```

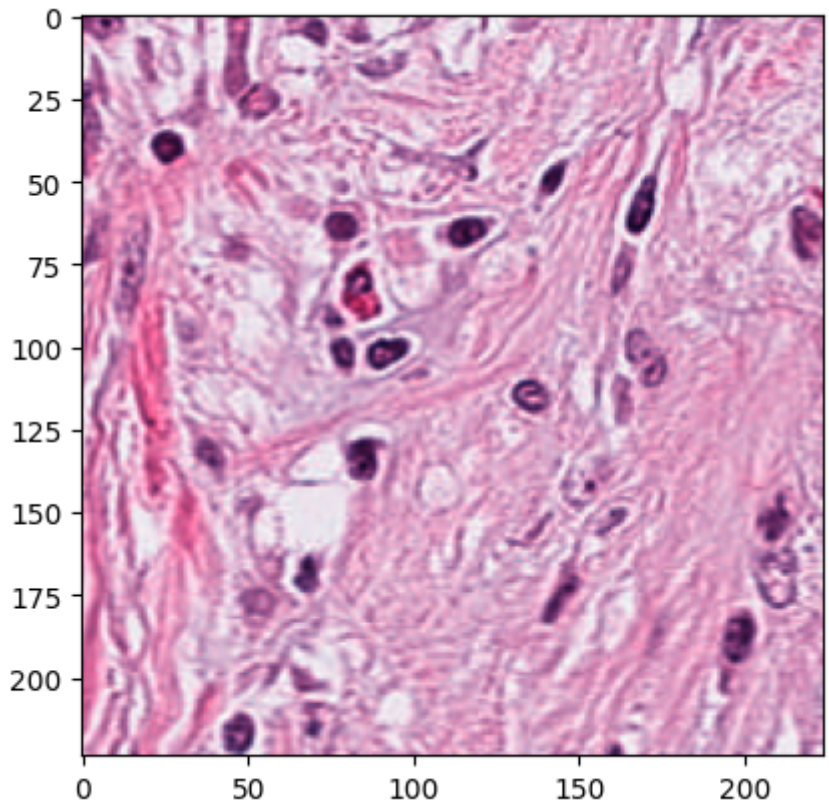
```
d_train_tiny = Dataset('train_tiny')


rand_i = np.random.randint(len(d_train_tiny))
image_tensor = d_train_tiny[rand_i][0]

# Assuming the tensor is in the correct format
image_numpy = image_tensor.permute(1, 2, 0).numpy().astype('uint8')
print(TISSUE_CLASSES[d_train_tiny[rand_i][1].item()])
# Display the image
```

```
11 plt.imshow(image_numpy)
12 plt.show()
```

```
Downloading...
From: https://drive.google.com/uc?export=download&confirm=pbef&id=1SxcuL7q7zO
To: /content/train_tiny.npz
100%|██████████| 105M/105M [00:00<00:00, 300MB/s]
Loading dataset train_tiny from npz.
Done. Dataset train_tiny consists of 900 images.
STR
```



Commited prev Dataset (because wrote own)

```
1 # d_train_tiny = Dataset('train_tiny')
2
3 # img, lbl = d_train_tiny.random_image_with_label()
4 # print()
5 # print(f'Got numpy array of shape {img.shape}, and label with code {lbl}.')
6 # print(f'Label code corresponds to {TISSUE_CLASSES[lbl]} class.')
7
8 # pil_img = Image.fromarray(img)
9 # IPython.display.display(pil_img)
```

LB 3

Data loader for NN

```
def get_dataloader(dataset, name):

  batch_size = 28
  if 'train' in name:
    aug = A.Compose([                                   # использую аугментацию
        A.RandomHorizontalFlip(p=0.5),
        A.RandomVerticalFlip(p=0.5),
        A.RandomRotation(degrees=30),
        A.RandomResizedCrop(224),                       # обрезка изображения
        A.ToTensor(),
        A.Normalize(mean=[187.9071, 135.8079, 179.5519], std=[32.6107, 41.3495
    ])
    #dataset = Dataset(name)
    loader = DataLoader(
        dataset=dataset,
        batch_size=batch_size,
        shuffle=True,                 # Перемешиваем для новых батчей
        num_workers=2,
        drop_last=True,
    )
    return loader
  else:
    aug = A.Compose([
        A.ToTensor(),
        A.Normalize(mean=[187.9071, 135.8079, 179.5519], std=[32.6107, 41.3495
    ])
    #dataset = Dataset(name)

    loader = DataLoader(
        dataset=dataset,
        batch_size=batch_size,
        shuffle=False,
        num_workers=2,
        drop_last=False,
    )
    return loader
```

Как вывел mean и std для нормализации

```
d_train_tiny = Dataset('train_tiny')
train_loader = get_dataloader(d_train_tiny, 'train_tiny')

mean = 0.
std = 0.
total_samples = 0

# Вычислите средние и стандартные значения для каждого канала
for images, _ in train_loader:
    images = images
    batch_samples = images.size(0)
    images = images.view(batch_samples, images.size(1), -1)
    mean += images.mean(2).sum(0)
```

```
    std += images.std(2).sum(0)
    total_samples += batch_samples

mean /= total_samples
std /= total_samples

print("Средние значения по каналам:", mean)
print("Стандартные отклонения по каналам:", std)
```

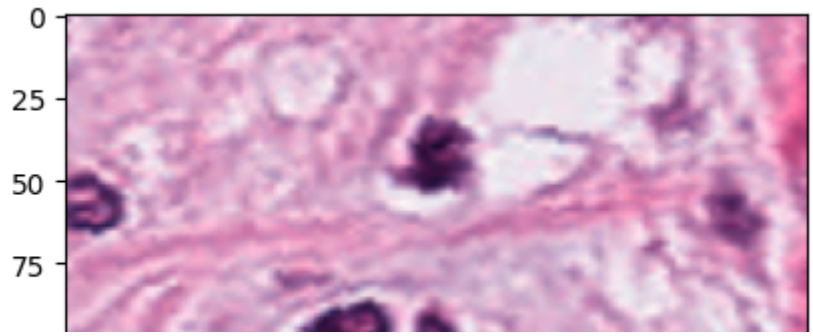
```
Downloading...
From: https://drive.google.com/uc?export=download&confirm=pbef&id=1SxcuL7q7zO
To: /content/train_tiny.npz
100%|██████████| 105M/105M [00:00<00:00, 258MB/s]
Loading dataset train_tiny from npz.
Done. Dataset train_tiny consists of 900 images.
/usr/local/lib/python3.10/dist-packages/torchvision/transforms/v2/_deprecated
  warnings.warn(
Средние значения по каналам: tensor([188.1168, 136.0152, 179.7652])
Стандартные отклонения по каналам: tensor([32.6554, 41.3942, 30.6649])
```

Пример изображения из LB2 с выполненной аугментацией

```
d_train_tiny[rand_i][0]


augmentation = transforms.Compose([
    A.ToTensor(),
    A.RandomHorizontalFlip(),  # Случайное отражение по горизонтали
    A.RandomRotation(degrees=30),  # Случайный поворот на угол до 30 градусов
    A.RandomResizedCrop(224),  # Случайное масштабирование и обрезка
    A.RandomVerticalFlip(),
    A.Normalize(mean=[187.9071, 135.8079, 179.5519], std=[32.6107, 41.3495, 30
      # Преобразование в тензор
])

augmented_image = augmentation(d_train_tiny[rand_i][0])
normalize_back = A.Normalize(mean=[-187.9071/32.6107, -135.8079/41.3495, -179.
plt.imshow((normalize_back(augmented_image).permute((1, 2, 0))/255).numpy())
plt.show()
```



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 len
        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(
```

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.

LB 4

```
1 !pip install wandb
```

```
Collecting wandb
  Downloading wandb-0.16.0-py3-none-any.whl (2.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 15.2 MB/s eta 0:00:0
Requirement already satisfied: Click!=8.0.0,>=7.1 in /usr/local/lib/python3.1
Collecting GitPython!=3.1.29,>=1.0.0 (from wandb)
  Downloading GitPython-3.1.40-py3-none-any.whl (190 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 190.6/190.6 kB 17.4 MB/s eta 0:
Requirement already satisfied: requests<3,>=2.0.0 in /usr/local/lib/python3.1
Requirement already satisfied: psutil>=5.0.0 in /usr/local/lib/python3.10/dis
Collecting sentry-sdk>=1.0.0 (from wandb)
  Downloading sentry_sdk-1.37.1-py2.py3-none-any.whl (251 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 251.7/251.7 kB 21.9 MB/s eta 0:
Collecting docker-pycreds>=0.4.0 (from wandb)
  Downloading docker_pycreds-0.4.0-py2.py3-none-any.whl (9.0 kB)
Requirement already satisfied: PyYAML in /usr/local/lib/python3.10/dist-packa
Collecting setproctitle (from wandb)
  Downloading setproctitle-1.3.3-cp310-cp310-manylinux_2_5_x86_64.manylinux1_
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: appdirs>=1.4.3 in /usr/local/lib/python3.10/di
Requirement already satisfied: protobuf!=4.21.0,<5,>=3.19.0 in /usr/local/lib
```

```
Requirement already satisfied: six>=1.4.0 in /usr/local/lib/python3.10/dist-p
Collecting gitdb<5,>=4.0.1 (from GitPython!=3.1.29,>=1.0.0->wandb)
  Downloading gitdb-4.0.11-py3-none-any.whl (62 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.7/62.7 kB 9.0 MB/s eta 0:00:
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/pyt
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.1
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.1
Collecting smmap<6,>=3.0.1 (from gitdb<5,>=4.0.1->GitPython!=3.1.29,>=1.0.0->
  Downloading smmap-5.0.1-py3-none-any.whl (24 kB)
Installing collected packages: smmap, setproctitle, sentry-sdk, docker-pycred
Successfully installed GitPython-3.1.40 docker-pycreds-0.4.0 gitdb-4.0.11 sen
```

```
import wandb     # для построения графиков в реальном времени
```

```
wandb.login()       # Если не захотите использовать wandb, то можно нажать ctr
```

```
wandb: Logging into wandb.ai. (Learn how to deploy a W&B server locally: http
wandb: You can find your API key in your browser here: https://wandb.ai/autho
wandb: Paste an API key from your profile and hit enter, or press ctrl+c to q
wandb: Appending key for api.wandb.ai to your netrc file: /root/.netrc
True
```

```
from torch.optim.lr_scheduler import CosineAnnealingLR
```

```
class Model(nn.Module):

    def __init__(self):
        super(Model, self).__init__()
        self.model = models.resnet34(pretrained=True)  # использую претрейн
        self.model.fc = nn.Linear(512, 9)
        self.device = torch.device("cuda" if torch.cuda.is_available() else "c
        self.tmp = Metrics()

    def save(self, name):
        with open('/content/drive/MyDrive/HW_MSU/checkpoint.pth', "wb") as fp:
          torch.save(self.model.state_dict(), fp)


    def load(self, name: str):
        name_to_id_dict = {
            'best': '1pKz6y-8jkmPN1GFWaxEl3DFJ_TN1Zl5P'
        }
        output = 'checkpoint.pth'
        gdown.download(f'https://drive.google.com/uc?id=1pKz6y-8jkmPN1GFWaxEl3

        with open('./checkpoint.pth', "rb") as fp:
          state_dict = torch.load(fp)
        self.model.load_state_dict(state_dict)

    def train(self, dt_train):
```

```
27
28          wandb.init(
29              # set the wandb project where this run will be logged
30              project="my-awesome-project_MSU",
31              name="adam + res34 - a2",
32              # track hyperparameters and run metadata
33              reinit=True,
34              config={
35                  "architecture": "CNN",
36                  "dataset": "From_MSU",
37                  "epochs": 21,
38              }
39          )
40
41          tmp = Metrics()
42
43          # первые 3 эпохи будем менять только последний слой
44          train_loader = get_dataloader(dt_train, 'train')
45          dt_test = Dataset('test')
46          val_dataloader = get_dataloader(dt_test, 'test')
47          loss_fn = nn.CrossEntropyLoss()               # ф-я потерь
48          for layer in list(self.model.children()):
49            layer.requires_grad_(False)            # морозим веса, т.к. будем сн
50          list(self.model.children())[-1].weight.requires_grad_(True)
51
52          optimizer = torch.optim.Adam(self.model.parameters(), lr=3e-4)
53          scheduler = CosineAnnealingLR(optimizer, T_max=int(len(train_loader) +
54          #device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
55
56          total_loss = 0
57          total_acc = 0
58          total_n = 0
59          self.model.train()
60          self.model.to(self.device)
61
62          for epoch in range(30):
63            if epoch > 3:
64              for layer in list(self.model.children()):
65                layer.requires_grad_(True)
66
67            total_loss = 0
68            total_loss_val = 0
69            self.model.train()
70            for batch_data, batch_labels in train_loader:
71              batch_data = batch_data.to(self.device)
72              batch_labels = batch_labels.to(self.device)
73              model_labels = self.model(batch_data)
74              # model_prediction = model.predict(batch_data)
75              new_loss = loss_fn(model_labels, batch_labels)
76
77              optimizer.zero_grad()
78              new_loss.backward()
79              optimizer.step()
80              scheduler.step()
81
```

```
            model_labels = model_labels.cpu().detach().numpy()
            model_labels = np.argmax(model_labels, axis=1)
            one_batch_loss = float(0.3 * tmp.accuracy(model_labels, batch_labe
            #print(one_batch_loss)
            wandb.log({"acc": one_batch_loss})      # логирование
            wandb.log({"lr": optimizer.param_groups[0]["lr"]})

            total_loss += one_batch_loss

          wandb.log({"epoch_acc": total_loss})

          self.save('/content/drive/MyDrive/HW_MSU/checkpoint.pth')

          self.val_test_on_dataset(data_loader=val_dataloader)




    def val_test_on_dataset(self, data_loader, limit=None):

        self.model.eval()  # Set the model to evaluation mode
        correct = 0
        total = 0
        total_loss_val = 0
        all_model_labels = []
        all_batch_labels = []

        with torch.no_grad():

            for batch_data, batch_labels in data_loader:
                batch_data = batch_data.to(self.device)
                batch_labels = batch_labels.to(self.device)
                model_labels = self.model(batch_data)
                model_labels = model_labels.cpu().detach().numpy()
                model_labels = np.argmax(model_labels, axis=1)
                # model_prediction = model.predict(batch_data)
                new_loss = float(0.3 * self.tmp.accuracy(model_labels, batch_l
                wandb.log({"acc_val": new_loss})
                all_model_labels.append(model_labels)
                all_batch_labels.append(batch_labels.cpu())

            all_model_labels = [item for sublist in all_model_labels for item
            all_batch_labels = [item for sublist in all_batch_labels for item
            total_loss_val = float(0.3 * self.tmp.accuracy(all_model_labels, a
            wandb.log({"epoch_acc_val_norm": total_loss_val})


    def test_on_dataset(self, dataset, limit=1):

        self.model.to(self.device)
        self.model.eval()  # Set the model to evaluation mode
        correct = 0
        total = 0
        total_loss_val = 0
        all_model_labels = []
```

```
        all_batch_labels = []
        data_loader = get_dataloader(dataset, 'test')

        with torch.no_grad():
          for batch_data, batch_labels in data_loader:
            batch_data = batch_data.to(self.device)
            batch_labels = batch_labels.to(self.device)
            model_labels = self.model(batch_data)
            model_labels = model_labels.cpu().detach().numpy()
            model_labels = np.argmax(model_labels, axis=1)
            # model_prediction = model.predict(batch_data)
            new_loss = float(0.3 * self.tmp.accuracy(model_labels, batch_label

            all_model_labels.append(model_labels)


          all_model_labels = [item for sublist in all_model_labels for item in
          ret_len = int(len(all_model_labels) * limit - 1)
          if limit == 1:
            return all_model_labels
          else:
            return all_model_labels[:ret_len]



    def test_on_image(self, img: np.ndarray):
        # todo: replace this code
        prediction = np.random.randint(9)
        sleep(0.05)
        return prediction
```

Идёт обучение на 30 эпох. И каждую эпоху сохраняем веса

```
model = Model()
d_train = Dataset('train')

model.train(d_train)
```

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

График accuracy на тестовом датасете можно посомтреть по ссылке: https://api.wandb.ai/links/airat-fayzullov-40/binpdr0i

График изменения learning rate: https://wandb.ai/airat-fayzullov-40/my-awesome-project_MSU/reports/lr-23-11-27-02-05-03---Vmlldzo2MDg1NDYw

---

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'.

```
1 d_train = Dataset('train_small')
2 d_test = Dataset('test_small')
```

```
Downloading...
From: https://drive.google.com/uc?export=download&confirm=pbef&id=1FosyL3b5rn
To: /content/train_small.npz
100%|██████████| 841M/841M [00:02<00:00, 300MB/s]
Loading dataset train_small from npz.
Done. Dataset train_small consists of 7200 images.
Downloading...
From: https://drive.google.com/uc?export=download&confirm=pbef&id=1vlgiKd69h_
To: /content/test_small.npz
100%|██████████| 211M/211M [00:00<00:00, 294MB/s]
Loading dataset test_small from npz.
Done. Dataset test_small consists of 1800 images.
```

```
1 model = Model()
2 if not EVALUATE_ONLY:
3     model.train(d_train)
```

```
    # model.save('best') это не нужно, т.к. сохраняю каждую эпоху
else:
    model.load('best')
```

```
/usr/local/lib/python3.10/dist-packages/torchvision/models/_utils.py:208: Use
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torchvision/models/_utils.py:223: Use
  warnings.warn(msg)
Downloading...
From (uriginal): https://drive.google.com/uc?id=1pKz6y-8jkmPN1GFWaxEl3DFJ_TN1
From (redirected): https://drive.google.com/uc?id=1pKz6y-8jkmPN1GFWaxEl3DFJ_T
To: /content/checkpoint.pth
100%|██████████| 85.3M/85.3M [00:00<00:00, 239MB/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)
Metrics.print_all(d_test.labels[:len(pred_1)], pred_1, '10% of test')
```

```
/usr/local/lib/python3.10/dist-packages/torchvision/transforms/v2/_deprecated
  warnings.warn(
metrics for 10% of test:
        accuracy 1.0000:
        balanced accuracy 1.0000:
```

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')
```

```
/usr/local/lib/python3.10/dist-packages/torchvision/transforms/v2/_deprecated
  warnings.warn(
AAAA
metrics for test:
        accuracy 0.9911:
        balanced accuracy 0.9911:
```

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