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/python3.12/dist-packages (5.2.0)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.12/dist-packages (from gdown) (4.13.5)
Requirement already satisfied: filelock in /usr/local/lib/python3.12/dist-packages (from gdown) (3.20.0)
Requirement already satisfied: requests[socks] in /usr/local/lib/python3.12/dist-packages (from gdown) (2.32.4)
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (from gdown) (4.67.1)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.12/dist-packages (from beautifulsoup4->gdow
Requirement already satisfied: typing-extensions>=4.0.0 in /usr/local/lib/python3.12/dist-packages (from beautiful
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests[
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests[socks]->gdow
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests[socks]
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests[socks]
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in /usr/local/lib/python3.12/dist-packages (from requests[sc
```

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 = False
TEST_ON_LARGE_DATASET = True
TISSUE_CLASSES = ('ADI', 'BACK', 'DEB', 'LYM', 'MUC', 'MUS', 'NORM', 'STR', 'TUM')
DATASETS_LINKS = {
    #'train': '1XtQzVQ5XbrfxpLHJuL0XBGJ5U7CS-cLi',
    'train': '1QmmI0A_QC6jnp8BqOYSJANhlyI54l1H0',
    'train_small': '1qd45xXfDwdZjktLFwQb-et-mAaFeCzOR',
    'train_tiny': '1I-2ZOuXLd4QwhZQQltp817Kn3J0Xgbui',
    #'test': '1RfPou3pFKpuHDJZ-D9XDFzgvwpUBFlDr',
    'test': '1t6xvmolT7T63v9yy0cu2ztCArJKxDTS3',
    '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
```

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]}"
            url = f'https://drive.google.com/uc?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 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.

↳ Скрыто 2 ячейки.

Обёртка над Dataset для использования с PyTorch

```
# ============================================================
# (Опционально) Обёртка над Dataset для использования с PyTorch
# ============================================================

# ВНИМАНИЕ:
# Этот код нужен только тем, кто хочет решать задание с помощью PyTorch.
# Он показывает, как "подключить" наш Dataset к torch.utils.data.DataLoader.

try:
    import torch
    from torch.utils.data import Dataset as TorchDataset, DataLoader
    import torchvision.transforms as T
    from PIL import Image

    class HistologyTorchDataset(TorchDataset):
        """
        Обёртка над Dataset для использования с PyTorch.

        base_dataset: экземпляр Dataset('train'), Dataset('train_small'), etc.
        transform:    функция/объект, преобразующий изображение (PIL.Image -> torch.Tensor).

        """
```

```
        def __init__(self, base_dataset, transform=None):
            self.base = base_dataset
            # Минимальный transform по умолчанию:
            # np.uint8 [0, 255] -> float32 [0.0, 1.0]
            self.transform = transform or T.ToTensor()

        def __len__(self):
            # Размер датасета
            return len(self.base.images)

        def __getitem__(self, idx):
            """
            Возвращает (image_tensor, label) для PyTorch.
            image_tensor: torch.Tensor формы [3, H, W]
            label: int
            """
            img, label = self.base.image_with_label(idx)  # img: np.ndarray (H, W, 3)
            img = Image.fromarray(img)                    # в PIL.Image
            img = self.transform(img)                     # в torch.Tensor
            return img, label

except ImportError:
    HistologyTorchDataset = None
    print("PyTorch / torchvision не найдены. Обёртка HistologyTorchDataset недоступна.")
```

Пример использвания класса HistologyTorchDataset

↳ Скрыта 1 ячейка.

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.

```
import numpy as np
import torch
import torch.nn as nn
import torch.optim as optim
from torchvision.models import resnet50, ResNet50_Weights
from torchvision import transforms as T
from torch.utils.data import DataLoader
from PIL import Image

from tqdm import tqdm
from sklearn.metrics import f1_score
import gdown
import matplotlib.pyplot as plt

class Model:

    def __init__(self):
        self.device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
        self.model = resnet50(weights=ResNet50_Weights.DEFAULT)
        self.model.fc = nn.Linear(self.model.fc.in_features, len(TISSUE_CLASSES))
        self.model.to(self.device)
        self.criterion = nn.CrossEntropyLoss()
        self.optimizer = torch.optim.Adam(self.model.parameters(), lr = 1e-4)
        self.epochs = 15
        self.best_f1 = 0.0

    def save(self, name: str):
        path = f"/content/drive/MyDrive/cmcmsu/{name}.pth"
        torch.save(self.model.state_dict(), path)
        print(f"Model saved to {path}")

    def load(self, name: str):
        name_to_id_dict = {
          'best': '1HNF1d-Vh9OBI3ghLtHAPkJiflo6l9s47',
          'latest': '1LX6VskPUgrvuuTmCfcjVDVK3FUrCJGnx'
        }
        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, map_location=self.device, weights_only=True))
        self.model = self.model.to(self.device)
        print(f"Model loaded from {output}")

    def train(self, dataset: Dataset):
        print(f'training started')
        #LBL11
        train_tf = T.Compose([
          T.Resize((224, 224)),
          T.RandomHorizontalFlip(),
          T.RandomVerticalFlip(),
          T.RandomRotation(25),
          T.ColorJitter(brightness=0.3, contrast=0.3, saturation=0.3),
          T.ToTensor(),
        ])
```

```
    val_tf = T.Compose([
      T.Resize((224, 224)),
      T.ToTensor(),
    ])

    #LBL1
    val_size = int(len(dataset.images) * 0.1)
    val_idx = np.random.choice(len(dataset.images), val_size, replace=False)
    train_idx = np.setdiff1d(np.arange(len(dataset.images)), val_idx)

    base_ds = HistologyTorchDataset(dataset)
    base_ds.transform = train_tf
    train_ds = torch.utils.data.Subset(base_ds, train_idx)
    val_ds = HistologyTorchDataset(dataset, transform=val_tf)
    val_ds = torch.utils.data.Subset(val_ds, val_idx)
    train_loader = DataLoader(train_ds, batch_size=32, shuffle=True)
    val_loader   = DataLoader(val_ds, batch_size=32, shuffle=False)

    losses = []
    f1_scores = []
    self.model.train()

    for epoch in range(1, self.epochs + 1):
      loss_sum = 0.0
      for xb, yb in train_loader:
        xb = xb.to(self.device)
        yb = yb.to(self.device)
        self.optimizer.zero_grad()
        logits = self.model(xb)
        loss = self.criterion(logits, yb)
        loss.backward()
        self.optimizer.step()
        loss_sum += loss.item()
      epoch_loss = loss_sum / len(train_loader)
      losses.append(epoch_loss)

      #LBL5
      print(f"[Epoch {epoch}/{self.epochs}], loss={epoch_loss:.4f}")
      f1 = self._validate_loader(val_loader)
      f1_scores.append(f1)
      print(f"[Epoch {epoch}] validation F1={f1:.4f}")

      #LBL3
      self.save('latest')
      if f1 > self.best_f1:
        self.best_f1 = f1
        self.save('best')
    print(f'training done')

    #LBL6
    plt.figure(figsize=(6,4))
    plt.plot(losses, marker='o')
    plt.title("Training Loss per Epoch (ResNet50)")
    plt.xlabel("Epoch")
    plt.ylabel("Loss")
    plt.grid(True)
    plt.show()

    plt.figure(figsize=(6, 4))
    plt.plot(f1_scores, marker='o', color='green')
    plt.title("Validation F1 per Epoch")
    plt.xlabel("Epoch")
    plt.ylabel("F1 Score")
    plt.grid(True)
    plt.show()

  def _validate_loader(self, val_loader):
    self.model.eval()
    preds, gts = [], []

    with torch.no_grad():
        for xb, yb in val_loader:
            xb = xb.to(self.device)
            logits = self.model(xb)
            y_pred = torch.argmax(logits, dim=1).cpu().numpy()
            preds.extend(y_pred)
```

```
                gts.extend(yb.numpy())

        return f1_score(gts, preds, average="macro")


    def test_on_dataset(self, dataset: Dataset, limit=None):
        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):
        self.model.eval()

        t = T.Compose([
            T.Resize((224, 224)),
            T.ToTensor(),
        ])

        pil = Image.fromarray(img)
        tensor = t(pil).unsqueeze(0).to(self.device)

        with torch.no_grad():
            logits = self.model(tensor)
            prediction = torch.argmax(logits, 1).item()

        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 (original): https://drive.google.com/uc?id=1QmmI0A_QC6jnp8BqOYSJANhlyI54l1H0
From (redirected): https://drive.google.com/uc?id=1QmmI0A_QC6jnp8BqOYSJANhlyI54l1H0&confirm=t&uuid=f06b33b2-0ef2-4
To: /content/train.npz
100%|██████████| 2.10G/2.10G [00:37<00:00, 56.7MB/s]
Loading dataset train from npz.
Done. Dataset train consists of 18000 images.
Downloading...
From (original): https://drive.google.com/uc?id=1t6xvmolT7T63v9yy0cu2ztCArJKxDTS3
From (redirected): https://drive.google.com/uc?id=1t6xvmolT7T63v9yy0cu2ztCArJKxDTS3&confirm=t&uuid=972c989a-8352-4
To: /content/test.npz
100%|██████████| 525M/525M [00:09<00:00, 55.1MB/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('latest')
else:
    #todo: your link goes here
    model.load('best')
```

```
Downloading: "https://download.pytorch.org/models/resnet50-11ad3fa6.pth" to /root/.cache/torch/hub/checkpoints/res
100%|██████████| 97.8M/97.8M [00:00<00:00, 187MB/s]
training started
[Epoch 1/15], loss=0.3418
[Epoch 1] validation F1=0.9794
Model saved to /content/drive/MyDrive/cmcmsu/latest.pth
Model saved to /content/drive/MyDrive/cmcmsu/best.pth
[Epoch 2/15], loss=0.2089
[Epoch 2] validation F1=0.9679
Model saved to /content/drive/MyDrive/cmcmsu/latest.pth
[Epoch 3/15], loss=0.1162
[Epoch 3] validation F1=0.9773
Model saved to /content/drive/MyDrive/cmcmsu/latest.pth
[Epoch 4/15], loss=0.0938
[Epoch 4] validation F1=0.9774
Model saved to /content/drive/MyDrive/cmcmsu/latest.pth
[Epoch 5/15], loss=0.0799
[Epoch 5] validation F1=0.9873
Model saved to /content/drive/MyDrive/cmcmsu/latest.pth
Model saved to /content/drive/MyDrive/cmcmsu/best.pth
```

Пример тестирования модели на части набора данных:

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

```
[Epoch 7/15], loss=0.0702
[Epoch 7] validation F1=0.9887
Model saved to /content/drive/MyDrive/cmcmsu/latest.pth
[Epoch 8/15], loss=0.0695
[Epoch 8] validation F1=0.9900
Model saved to /content/drive/MyDrive/cmcmsu/latest.pth
[Epoch 9/15], loss=0.0516
[Epoch 9] validation F1=0.9890
Model saved to /content/drive/MyDrive/cmcmsu/latest.pth
[Epoch 10/15], loss=0.0606
Model saved to /content/drive/MyDrive/cmcmsu/latest.pth
[Epoch 11/15], loss=0.0437
[Epoch 11] validation F1=0.9889
Model saved to /content/drive/MyDrive/cmcmsu/latest.pth
[Epoch 12/15], loss=0.0507
[Epoch 12] validation F1=0.9760
Model saved to /content/drive/MyDrive/cmcmsu/latest.pth
[Epoch 13/15], loss=0.0435
[Epoch 13] validation F1=0.9880
Model saved to /content/drive/MyDrive/cmcmsu/latest.pth
[Epoch 14/15], loss=0.0430
[Epoch 14] validation F1=0.9880
Model saved to /content/drive/MyDrive/cmcmsu/latest.pth
[Epoch 15/15], loss=0.0430
[Epoch 15] validation F1=0.9880
Model saved to /content/drive/MyDrive/cmcmsu/latest.pth
```

```
Downloading: "https://download.pytorch.org/models/resnet50-11ad3fa6.pth" to /root/.cache/torch/hub/checkpoints/res
100%|██████████| 97.8M/97.8M [00:01<00:00, 70.7MB/s]
Downloading: https://drive.google.com/uc?id=1HNF1d-Vh9OBI3ghLtHAPkJiflo6l9s47
From (original): https://drive.google.com/uc?id=1HNF1d-Vh9OBI3ghLtHAPkJiflo6l9s47
From (redirected): https://drive.google.com/uc?id=1HNF1d-Vh9OBI3ghLtHAPkJiflo6l9s47&confirm=t&uuid=23ba49e6-bd3e-4
To: /content/best.pth
100%|██████████| 94.4M/94.4M [00:02<00:00, 38.6MB/s]
Model loaded from best.pth
```

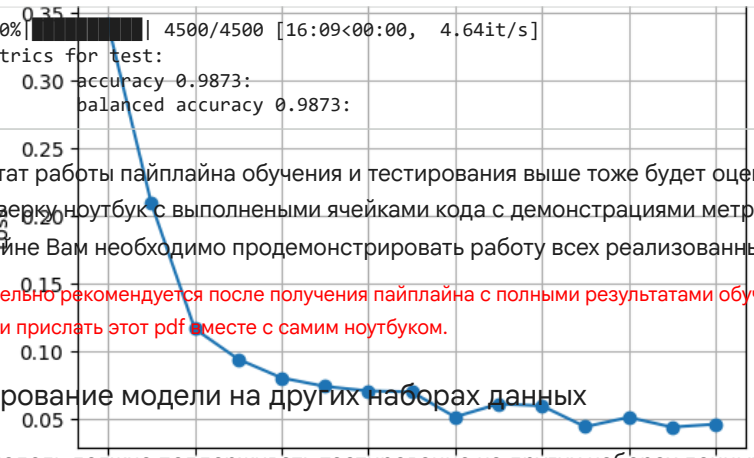
```
# evaluating model on 10% of test dataset
pred_1 = model.test_on_dataset(d_test, limit=0.1)
Metrics.print_all(d_test.labels[:len(pred_1)], pred_1, '10% of test')
```

```
100%|██████████| 4507450 [01:37<00:00,  4.59it/s]metrics for 10% of test:
        accuracy 0.9978:
        balanced accuracy 0.9978:
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:2524: UserWarning: y_pred contains clas
  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 [16:09<00:00,  4.64it/s]
metrics for test:
        accuracy 0.9873:
        balanced accuracy 0.9873:
```



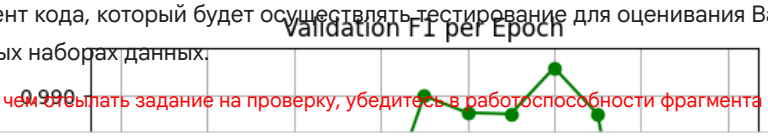
Результат работы пайплайна обучения и тестирования выше тоже будет оцениваться. Поэтому не забудьте присылать на проверку ноутбук с выполнеными ячейками кода с демонстрациями метрик обучения, графиками и т.п. В этом пайплайне Вам необходимо продемонстрировать работу всех реализованных дополнений, улучшений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

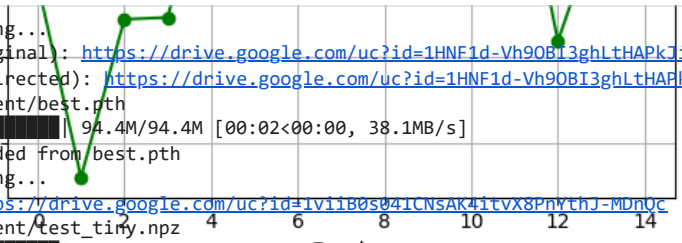
Прежде чем отсылать задание на проверку, убедитесь в работоспособности фрагмента кода ниже.




```
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 (original): https://drive.google.com/uc?id=1HNF1d-Vh9OBI3ghLtHAPkJiflo6l9s47
From (redirected): https://drive.google.com/uc?id=1HNF1d-Vh9OBI3ghLtHAPkJiflo6l9s47&confirm=t&uuid=2a727023-ec97-4
To: /content/best.pth
100%|██████████| 94.4M/94.4M [00:02<00:00, 38.1MB/s]
Model loaded from best.pth
Downloading...
From: https://drive.google.com/uc?id=1Vi1B0s04ICNsAK4itvX8PnYthJ-MDnOc
To: /content/test_tiny.npz
100%|██████████| 10.6M/10.6M [00:00<00:00, 60.9MB/s]
Loading dataset test_tiny from npz...
Done. Dataset test_tiny consists of 90 images.
100%|██████████| 90/90 [00:18<00:00,  4.83it/s]metrics for test-tiny:
        accuracy 0.9667:
        balanced accuracy 0.9667:
```



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:

↳ Скрыта 1 ячейка.

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:

↳ Скрыта 1 ячейка.

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.

↳ Скрыта 1 ячейка.

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.

↳ Скрыто 4 ячейки.

PyTorch

↳ Скрыта 1 ячейка.

Дополнительные ресурсы по PyTorch

- **Официальные туториалы PyTorch** — https://pytorch.org/tutorials/
- **"Deep Learning with PyTorch: 60-Minute Blitz"** — https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html
- **Transfer Learning for Computer Vision** — https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html
- **PyTorch Get Started (установка)** — https://pytorch.org/get-started/locally/
- **Dive into Deep Learning (D2L, глава PyTorch)** — https://d2l.ai/chapter_preliminaries/index.html
- **Fast.ai — Practical Deep Learning for Coders** — https://course.fast.ai/
- **Learn PyTorch.io (Zero to Mastery)** — https://www.learnpytorch.io/

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
!unzip -uq "tmp.zip" -d "tmp2"
```