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->gdown) (2.8)
Requirement already satisfied: typing-extensions>=4.0.0 in /usr/local/lib/python3.12/dist-packages (from beautifulsoup4->gdc
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests[socks]->gd
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests[socks]->gdown) (3.11)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests[socks]->gdown) (
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests[socks]->gdown) (
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in /usr/local/lib/python3.12/dist-packages (from requests[socks]->gdow
```

Монтирование Вашего Google Drive к текущему окружению:

```
from google.colab import drive
drive.mount('/content/drive', force_remount=True)
```

```
Mounted at /content/drive
```

```
!ls /content/drive/MyDrive
```

```
 1
 122
 1.tif
 2.tif
 ACF.gsheet
 ACFoverdues.gsheet
'Adobe Scan Dec 18, 2020.pdf'
'Adobe Scan Jan 19, 2021.pdf'
'Adobe Scan Oct 27, 2020.pdf'
 akbar.csv
 akbar.gsheet
 base
 base.csv
 best_v1.3_kaggle
 CLEANED_RETURNS.csv
'Colab Notebooks'
 data_2009-2024.csv
 FORTE.gslides
'IELTS Practice Test Plus.pdf'
 IMG_20190824_0001.pdf
 IMG_20190824_0002.pdf
 lougheed_lin_barron_s_writing_for_the_ielts.pdf
 period2.csv
 Peryshkin_Aleksandr_Fizika._7_klass_uchebnik_dlya_obsheobrazovatelnyh_uchrechdenii_Litmir.net_270397_original_6c2c0.pdf
 reading_Polnyy.gdoc
 reed.pdf
'Scan Oct 17, 2020.pdf'
'Temirlan 26 October'
'Temirlan Essays'
'Temirlan Narembekov Early Admission 2020-2021'
'Temirlan Narembekov Regular Decision 2020-2021'
'The Official Cambridge Guide to IELTS. Student’s Book with Answers with DVD-ROM ( PDFDrive.com ).pdf'
'Ulichnaya-Prestupnost-v-Kazahstane-Obzor-2025-goda (1).gslides'
'АКФ 22-03-2023 Мехмат ШАБЛОН.xlsx'
'Без названияреr'
'Исследование ценовых рядов на основании машинного обучения.gslides'
'Копия antifraud DS.gsheet'
 Мо
'Нарембеков Т Кандидат .gsheet'
 Основы_параллельного_программирования.pdf
'Патент на 2018 год 1 полугодие.pdf'
 ПТД-Форма.pdf
'синильная кислота.gslides'
' Системы международных отношений  и крупные международные организации  (краткая историческая справка, сравнительный анализ)
'ССылкиНа Видео ЧМ_ed430e8e-7f05-4f45-8393-1716504785aa (1).gdoc'
'ССылкиНа Видео ЧМ_ed430e8e-7f05-4f45-8393-1716504785aa (2).gdoc'
'ССылкиНа Видео ЧМ_ed430e8e-7f05-4f45-8393-1716504785aa.gdoc'
'Темирлан Нарембеков Свободный Таргет'
```

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_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
```

Класс Dataset

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

```
class Dataset:

    def __init__(self, name):
        self.name = name
        self.is_loaded = False
        url = f"https://drive.google.com/uc?export=download&confirm=pbef&id={DATASETS_LINKS[name]}"
        output = f'{name}.npz'
        gdown.download(url, output, quiet=False)
        print(f'Loading dataset {self.name} from npz.')
        np_obj = np.load(f'{name}.npz')
        self.images = np_obj['data']
        self.labels = np_obj['labels']
        self.n_files = self.images.shape[0]
        self.is_loaded = True
        print(f'Done. Dataset {name} consists of {self.n_files} images.')

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

    def images_seq(self, n=None):
        # sequential access to images inside dataset (is needed for testing)
        for i in range(self.n_files if not n else n):
            yield self.image(i)

    def random_image_with_label(self):
        # get random image with label from dataset
        i = np.random.randint(self.n_files)
        return self.image(i), self.labels[i]

    def random_batch_with_labels(self, n):
        # create random batch of images with labels (is needed for training)
        indices = np.random.choice(self.n_files, n)
        imgs = []
        for i in indices:
            img = self.image(i)
            imgs.append(self.image(i))
        logits = np.array([self.labels[i] for i in indices])
        return np.stack(imgs), logits

    def image_with_label(self, i: int):
        # return i-th image with label from dataset
        return self.image(i), self.labels[i]
```

Пример 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.

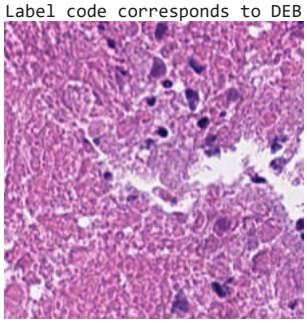
```
d_train_tiny = Dataset('train')

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=1XtQzVQ5XbrfxpLHJuL0XBGJ5U7CS-cLi
To: /kaggle/working/train.npz
100%|██████████| 2.10G/2.10G [00:20<00:00, 104MB/s]
Loading dataset train from npz.
Done. Dataset train consists of 18000 images.

Got numpy array of shape (224, 224, 3), and label with code 2.
Label code corresponds to DEB class.
```



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

```
if "HistologyTorchDataset" not in globals() or HistologyTorchDataset is None:
    print("PyTorch не установлен или обёртка недоступна — пример пропущен.")
else:
    print("Пример использования PyTorch-обёртки над Dataset")

    base_train = Dataset('train_tiny')

    # Создаём PyTorch-совместимый датасет
    train_ds = HistologyTorchDataset(base_train)

    # DataLoader автоматически создаёт батчи и перемешивает данные
    from torch.utils.data import DataLoader
    train_loader = DataLoader(train_ds, batch_size=8, shuffle=True)

    # Берём один батч и выводим информацию
    images_batch, labels_batch = next(iter(train_loader))

    print("Форма батча изображений:", tuple(images_batch.shape))  # [batch, 3, 224, 224]
    print("Форма батча меток:", tuple(labels_batch.shape))        # [batch]
    print("Пример меток:", labels_batch[:10].tolist())

    print("Тип images_batch:", type(images_batch))
    print("Тип labels_batch:", type(labels_batch))
```

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.

```
from time import sleep
import os
import torch.nn as nn
import torch.nn.functional as F
class Model(nn.Module):
    WEIGHTS_URL = "https://drive.google.com/uc?id=169J4gR-hcfdGYAKnJ-XuMzF_kC6UmNnn"
    WEIGHTS_LOCAL_PATH = "best_v1.3_kaggle.pth"  # как хочешь назвать локальный файл
    def __init__(self, input_shape = (3,224,224)):
        super().__init__()
        self.conv1 = nn.Conv2d(in_channels = 3, out_channels = 32, kernel_size = 3)
        self.pool = nn.MaxPool2d(kernel_size = 2, stride = 2)
        self.conv2 = nn.Conv2d(in_channels = 32, out_channels = 64, kernel_size=3)
        self.conv3 = nn.Conv2d(in_channels = 64, out_channels = 64, kernel_size = 3)
        self.conv4 = nn.Conv2d(in_channels = 64, out_channels = 64, kernel_size = 3)
        self.conv5 = nn.Conv2d(in_channels = 64, out_channels = 64, kernel_size = 3)
        #self.conv6 = nn.Conv2d(in_channels = 64, out_channels = 64, kernel_size = 3)
        flatten_dim = self._compute_flatten_dim(input_shape)
        #self.flatten = nn.Flatten()
        self.fc1=nn.Linear(flatten_dim, 64)
        self.fc2=nn.Linear(64, len(TISSUE_CLASSES))

    def forward(self, x):
        x = self._forward_features(x)
        x = x.view(x.size(0),-1)
        x = F.relu(self.fc1(x))
        logits = self.fc2(x)
        return logits

    def _forward_features(self,x):
        x = F.relu(self.conv1(x))
        x = self.pool(x)
        x = F.relu(self.conv2(x))
        x = self.pool(x)
        x = F.relu(self.conv3(x))
        x = self.pool(x)
        x = F.relu(self.conv4(x))
        x = self.pool(x)
        x = F.relu(self.conv5(x))
        x = self.pool(x)
        return x

    def _compute_flatten_dim(self, input_shape):
        c, h, w = input_shape
        with torch.no_grad():
          dummy = torch.zeros(1, c, h, w)
          x = self._forward_features(dummy)
          flatten_dim = x.view(1, -1).shape[1]
        return flatten_dim

    def fit(self, dataset, epochs=10, batch_size = 64, lr=1e-3, device = None):
        # you can add some plots for better visualization,
        # you can add model autosaving during training,
        # etc.
        print(f'training started')
        if device is None:
            device = next(self.parameters()).device
        self.to(device)
        dataloader = DataLoader(dataset, batch_size=batch_size, shuffle=True)
        optimizer = torch.optim.Adam(self.parameters(), lr=lr)
```

```
        criterion = nn.CrossEntropyLoss()

        for epoch in range(epochs):
          self.train()
          running_loss = 0.0
          correct = 0
          total = 0

          for images, labels in dataloader:
              images = images.to(device)
              labels = labels.to(device)

              optimizer.zero_grad()

              logits = self(images)
              loss = criterion(logits,labels)

              loss.backward()
              optimizer.step()


              running_loss += loss.item() * images.size(0)
              _,preds = torch.max(logits,dim=1)
              total += labels.size(0)
              correct += (preds == labels).sum().item()
          epoch_loss = running_loss / total
          epoch_acc = correct / total
          print(f"Epoch {epoch+1}/{epochs} | loss={epoch_loss:.4f} | acc={epoch_acc:.4f}")



        sleep(2)
        print(f'training done')
        pass

    def test_on_image(self, img, device=None):
        """
        img может быть:
          - torch.Tensor (скорее всего так и есть, если в Dataset используется ToTensor)
          - np.ndarray (если ты где-то сам читаешь картинки через cv2/plt и т.п.)
        """
        if device is None:
            device = next(self.parameters()).device

        self.eval()
        with torch.no_grad():
            # 1. Приводим к torch.Tensor
            if isinstance(img, torch.Tensor):
                x = img.float()
            elif isinstance(img, np.ndarray):
                x = torch.from_numpy(img).float()
            else:
                raise TypeError(f"Ожидался Tensor или np.ndarray, а пришло {type(img)}")

            # 2. Приводим к формату (3, H, W)
            if x.ndim == 3:
                # если (H, W, 3), то делаем (3, H, W)
                if x.shape[0] != 3 and x.shape[-1] == 3:
                    x = x.permute(2, 0, 1)
            elif x.ndim == 2:
                # на случай ЧБ, можно расширить до (1, H, W) — но у тебя in_channels=3
                raise ValueError(f"Ожидался 3-канальный вход, а пришла форма {x.shape}")

            # 3. Нормализация, если вдруг значения в [0, 255]
            if x.max() > 1.5:
                x = x / 255.0

            # 4. Добавляем batch dimension
            if x.ndim == 3:
                x = x.unsqueeze(0)   # (1, 3, H, W)

            x = x.to(device)

            # 5. Прогон через сеть
            logits = self(x)
            probs = F.softmax(logits, dim=1)
            pred = torch.argmax(probs, dim=1).item()

        return pred

    def test_on_dataset(self, dataset, limit=None, device=None):
        # берём девайс из модели
        if device is None:
```

```
            device = next(self.parameters()).device

        self.to(device)
        self.eval()
        predictions = []

        # --- Вариант 1: "старый" Dataset из задания (d_test)
        # у него есть n_files и images_seq
        if hasattr(dataset, "n_files") and hasattr(dataset, "images_seq"):
            total = dataset.n_files
            n = total if limit is None else int(total * limit)

            for img in tqdm(dataset.images_seq(n), total=n):
                # img тут, скорее всего, np.ndarray
                pred = self.test_on_image(img, device=device)
                predictions.append(pred)

        # --- Вариант 2: PyTorch Dataset (HistologyTorchDataset)
        # у него есть __len__ и __getitem__
        else:
            try:
                total = len(dataset)
            except TypeError:
                raise TypeError(
                    "dataset не поддерживает ни n_files/images_seq, ни len()/__getitem__. "
                    "Убедись, что передаёшь либо d_test, либо PyTorch Dataset."
                )

            n = total if limit is None else int(total * limit)

            for i in tqdm(range(n), total=n):
                img, _ = dataset[i]      # (image, label)
                pred = self.test_on_image(img, device=device)
                predictions.append(pred)

        return predictions

    def save(self, path: str = "best_model.pt"):
        torch.save(self.state_dict(), path)
        print(f"Модель сохранена в {path}")

    def load(self, path=None):
        """
        Автоматически загружает веса обученной модели из облачного хранилища (Google Drive),
        если локального файла нет.
        Если path не задан, использует WEIGHTS_LOCAL_PATH.
        """
        if path is None:
            path = self.WEIGHTS_LOCAL_PATH

        # если файла нет — качаем из облака
        if not os.path.exists(path):
            print(f"[Model.load] Файл '{path}' не найден, скачиваю из Google Drive...")
            try:
                import gdown
            except ImportError:
                import sys, subprocess
                subprocess.check_call([sys.executable, "-m", "pip", "install", "-q", "gdown"])
                import gdown

            # скачиваем
            gdown.download(self.WEIGHTS_URL, path, quiet=False)

        # загружаем веса
        state_dict = torch.load(path, map_location="cpu")
        self.load_state_dict(state_dict)
        self.eval()
        print("[Model.load] Веса успешно загружены.")
```

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')
```

```
train_ds = HistologyTorchDataset(d_train)          # ToTensor по умолчанию
test_ds  = HistologyTorchDataset(d_test)
```

```
Downloading...
From: https://drive.google.com/uc?export=download&confirm=pbef&id=1XtQzVQ5XbrfxpLHJuL0XBGJ5U7CS-cLi
To: /content/train.npz
100%|██████████| 2.10G/2.10G [00:34<00:00, 61.8MB/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=1RfPou3pFKpuHDJZ-D9XDFzgvwpUBFlDr
To: /content/test.npz
100%|██████████| 525M/525M [00:09<00:00, 53.3MB/s]
Loading dataset test from npz.
Done. Dataset test consists of 4500 images.
```

```
model = Model()
if not EVALUATE_ONLY:
    model.fit(train_ds)
    model.save('best_v1.3_kaggle')
else:
    #todo: your link goes here
    model.load('best_v1.3_kaggle')
```

```
training started
Epoch 1/10 | loss=1.9635 | acc=0.2444
Epoch 2/10 | loss=1.1494 | acc=0.5526
Epoch 3/10 | loss=0.7958 | acc=0.6949
Epoch 4/10 | loss=0.6299 | acc=0.7634
Epoch 5/10 | loss=0.5092 | acc=0.8115
Epoch 6/10 | loss=0.4589 | acc=0.8340
Epoch 7/10 | loss=0.4141 | acc=0.8523
Epoch 8/10 | loss=0.3770 | acc=0.8671
Epoch 9/10 | loss=0.3255 | acc=0.8854
Epoch 10/10 | loss=0.2882 | acc=0.8987
training done
Модель сохранена в best_v1.3_kaggle
```

```
from IPython.display import FileLink
FileLink('best_v1.3_kaggle')
```

best_v1.3_kaggle

```
# # старый обученный объект
# old_model = model

# # новый объект — уже НОВОГО класса Model
# new_model = Model()

# # переносим веса
# new_model.load_state_dict(old_model.state_dict())

# # теперь работать будем с new_model
# model = new_model
```

Пример тестирования модели на части набора данных:

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

```
  0%|          | 0/450 [00:00<?, ?it/s]
metrics for 10% of test:
     accuracy 0.9200:
     balanced accuracy 0.9200:
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:2184: UserWarning: y_pred contains classes not ir
  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')
```

```
  0%|          | 0/4500 [00:00<?, ?it/s]
metrics for test:
     accuracy 0.8864:
     balanced accuracy 0.8864:
```

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.

```
model_path = "/content/drive/MyDrive/best_v1.3_kaggle"
```

```
final_model = Model()
final_model.load()
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')
```

```
[Model.load] Файл 'best_v1.3_kaggle.pth' не найден, скачиваю из Google Drive...
Downloading...
From: https://drive.google.com/uc?id=169J4gR-hcfdGYAKnJ-XuMzF_kC6UmNnn
To: /content/best_v1.3_kaggle.pth
100%|██████████| 937k/937k [00:00<00:00, 81.3MB/s]
[Model.load] Веса успешно загружены.
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, 98.3MB/s]
Loading dataset test_tiny from npz.
Done. Dataset test_tiny consists of 90 images.
100%                                                  90/90 [00:05<00:00, 23.01it/s]
metrics for test-tiny:
     accuracy 0.8778:
     balanced accuracy 0.8778:
```

```
final_model = Model()
final_model.load()
d_test_small = Dataset('test_small')
pred = final_model.test_on_dataset(d_test_small)
Metrics.print_all(d_test_small.labels, pred, 'test-small')
```

```
[Model.load] Веса успешно загружены.
Downloading...
From: https://drive.google.com/uc?export=download&confirm=pbef&id=1wbRsog0n7uGlHIPGLhyN-PMeT2kdQ2lI
To: /content/test_small.npz
100%|██████████| 211M/211M [00:01<00:00, 148MB/s]
Loading dataset test_small from npz.
Done. Dataset test_small consists of 1800 images.
100%                                                  1800/1800 [00:54<00:00, 33.79it/s]
metrics for test-small:
     accuracy 0.8833:
     balanced accuracy 0.8833:
```

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.metrics import ConfusionMatrixDisplay
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)))

# Матрица ошибок + визуализация в новом API
cm = metrics.confusion_matrix(y_test, predicted)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
                              display_labels=classifier.classes_)

fig, ax = plt.subplots(figsize=(4, 4))
disp.plot(ax=ax)
```

```
ax.set_title("Confusion Matrix")
plt.tight_layout()
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.

PyTorch

```
# ============================================================
# Дополнительно: мини-демо PyTorch
# ============================================================
# Ранний выход, если PyTorch/обёртка недоступны
try:
    import torch
    import torch.nn as nn
    import torch.nn.functional as F
    from torch.utils.data import DataLoader
except ImportError:
    print("PyTorch не установлен — демо пропущено.")
    import sys
    raise SystemExit

if "HistologyTorchDataset" not in globals() or HistologyTorchDataset is None:
    print("HistologyTorchDataset недоступна — демо пропущено.")
    import sys
    raise SystemExit


# --- Данные: tiny-наборы, чтобы выполнялось быстро ---
base_train = Dataset('train_tiny')
base_test  = Dataset('test_tiny')

train_ds = HistologyTorchDataset(base_train)           # ToTensor по умолчанию
test_ds  = HistologyTorchDataset(base_test)

train_loader = DataLoader(train_ds, batch_size=16, shuffle=True)
test_loader  = DataLoader(test_ds,  batch_size=32, shuffle=False)

# --- Мини-модель: двухслойный CNN + один FC (демонстрация, не решение) ---
class TinyCNN(nn.Module):
    def __init__(self, num_classes=len(TISSUE_CLASSES)):
        super().__init__()
        self.conv1 = nn.Conv2d(3, 8, kernel_size=3, padding=1)
        self.conv2 = nn.Conv2d(8, 16, kernel_size=3, padding=1)
        self.pool  = nn.MaxPool2d(2, 2)  # 224->112->56
        self.fc    = nn.Linear(16 * 56 * 56, num_classes)

    def forward(self, x):
```

```
            x = self.pool(F.relu(self.conv1(x)))  # [B, 3, 224, 224] -> [B, 8, 112, 112]
            x = self.pool(F.relu(self.conv2(x)))  # [B, 8, 112, 112] -> [B, 16, 56, 56]
            x = x.view(x.size(0), -1)
            x = self.fc(x)
            return x

    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    print("Device:", device)

    model = TinyCNN(num_classes=len(TISSUE_CLASSES)).to(device)
    criterion = nn.CrossEntropyLoss()
    optimizer = torch.optim.Adam(model.parameters(), lr=1e-3)

    # --- Один учебный шаг "обучения" на одном батче ---
    model.train()
    xb, yb = next(iter(train_loader))
    xb = xb.to(device)
    yb = yb.to(device, dtype=torch.long)

    optimizer.zero_grad()
    logits = model(xb)
    loss = criterion(logits, yb)
    float_loss = float(loss.detach().cpu())
    loss.backward()
    optimizer.step()

    print(f"Loss на одном батче train_tiny: {float_loss:.4f}")

    # --- Быстрая проверка на одном батче теста (для формы вывода/метрик) ---
    model.eval()
    with torch.no_grad():
        xt, yt = next(iter(test_loader))
        xt = xt.to(device)
        logits_t = model(xt).cpu()
        y_pred = logits_t.argmax(dim=1).numpy()
        y_true = yt.numpy()

    print("Размерности:", {"y_true": y_true.shape, "y_pred": y_pred.shape})
    Metrics.print_all(y_true, y_pred, "_")  # balanced accuracy/accuracy на одном батче для демонстрации

    # для полноценного решения требуется собственный тренировочный цикл по эпохам,
    # аугментации/нормализация, сохранение/загрузка весов, и тестирование на всём наборе.
```

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"
```