Практическое задание №1

Установка необходимых пакетов:

```
#!pip install -q tqdm
#!pip install --upgrade --no-cache-dir gdown
```

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_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):
```

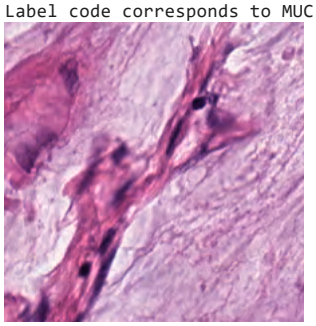
```
            yield self.image(i)

    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_tiny')

img, lbl = d_train_tiny.random_image_with_label()
print()
print(f'Got numpy array of shape {img.shape}, and label with code {lbl}.')
print(f'Label code corresponds to {TISSUE_CLASSES[lbl]} class.')

pil_img = Image.fromarray(img)
IPython.display.display(pil_img)
```

```
Downloading...
From: https://drive.google.com/uc?export=download&confirm=pbef&id=1I-2ZOuXLd4QwhZQQltp817Kn3J0Xgbui
To: /content/train_tiny.npz
100%|██████████| 105M/105M [00:00<00:00, 111MB/s]
Loading dataset train_tiny from npz.
Done. Dataset train_tiny consists of 900 images.

Got numpy array of shape (224, 224, 3), and label with code 4.
Label code corresponds to MUC 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))
```

```
Пример использования PyTorch-обёртки над Dataset
Downloading...
From: https://drive.google.com/uc?export=download&confirm=pbef&id=1I-2ZOuXLd4QwhZQQltp817Kn3J0Xgbui
To: /content/train_tiny.npz
100%|██████████| 105M/105M [00:00<00:00, 127MB/s]
Loading dataset train_tiny from npz.
Done. Dataset train_tiny consists of 900 images.
Форма батча изображений: (8, 3, 224, 224)
Форма батча меток: (8,)
Пример меток: [3, 4, 8, 4, 6, 2, 6, 6]
Тип images_batch: <class 'torch.Tensor'>
Тип labels_batch: <class 'torch.Tensor'>
```

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 tqdm import tqdm
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, classification_report
import itertools


# label 1
class WSConv2d(nn.Conv2d):
    def forward(self, x):
        w = self.weight
        w = (w - w.mean([1,2,3], keepdim=True)) / (w.std([1,2,3], keepdim=True) + 1e-5)
        return nn.functional.conv2d(
            x, w, self.bias, self.stride, self.padding, self.dilation, self.groups
        )


# label 2
class ConvBlock(nn.Module):
    def __init__(self, in_ch, out_ch, dropout=0.05):
        super().__init__()
        self.block = nn.Sequential(
```

```
            WSConv2d(in_ch, out_ch, kernel_size=3, padding=1),
            nn.BatchNorm2d(out_ch),
            nn.GELU(),
            nn.MaxPool2d(2),
            nn.Dropout(dropout)
        )

    def forward(self, x):
        return self.block(x)


# label 3
class Model:
    def __init__(self, num_classes: int = 9):
        # label 3.1
        self.device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
        self.num_classes = num_classes

        # label 3.2
        self.net = nn.Sequential(
            ConvBlock(3, 40, dropout=0.05),
            ConvBlock(40, 80, dropout=0.05),
            ConvBlock(80, 160, dropout=0.10),

            nn.AdaptiveAvgPool2d((1, 1)),
            nn.Flatten(),

            nn.Linear(160, 256),
            nn.GELU(),
            nn.Dropout(0.35),
            nn.LayerNorm(256),

            nn.Linear(256, self.num_classes)
        ).to(self.device)

        # label 3.3
        self.loss_fn = nn.CrossEntropyLoss(label_smoothing=0.1)
        self.optimizer = optim.Adam(self.net.parameters(), lr=1e-3, weight_decay=1e-4)

        self.scheduler = optim.lr_scheduler.ReduceLROnPlateau(
            self.optimizer, mode="min", factor=0.5, patience=2
        )

    # label 4
    def save(self, name: str):
        path = f'/content/drive/MyDrive/{name}.pth'
        torch.save(self.net.state_dict(), path)
        print(f"Model saved to {path}")

    def load(self, name: str):
        path = f'/content/drive/MyDrive/{name}.pth'
        state = torch.load(path, map_location=self.device)
        self.net.load_state_dict(state)
        self.net.eval()
        print(f"Model loaded from {path}")

    # label 5
    def apply_maskout(self, t: torch.Tensor, p: float = 0.5, size_frac: float = 0.3):
        if torch.rand(1).item() >= p:
            return t

        _, _, h, w = t.shape
        block = int(size_frac * min(h, w))
        if block < 1:
            return t

        pos_x = torch.randint(0, w, (1,)).item()
        pos_y = torch.randint(0, h, (1,)).item()

        x1, x2 = max(pos_x - block//2, 0), min(pos_x + block//2, w)
        y1, y2 = max(pos_y - block//2, 0), min(pos_y + block//2, h)

        t[:, :, y1:y2, x1:x2] = 0.0
        return t

    # label 6
    def prepare_tensor(self, img: np.ndarray, is_train: bool = False) -> torch.Tensor:
        if img.ndim == 2:
            img = np.stack([img, img, img], axis=-1)
        if img.shape[0] == 3:
            img = np.transpose(img, (1, 2, 0))

        img = img.astype(np.float32)
```

```
        img = img.astype(np.float32)
        if img.max() > 1:
            img /= 255.0

        img = (img - 0.5) / 0.5
        img = np.transpose(img, (2, 0, 1))

        t = torch.tensor(img, dtype=torch.float32).unsqueeze(0)

        if is_train:
            if torch.rand(1).item() < 0.5:
                t = torch.flip(t, dims=[3])

            if torch.rand(1).item() < 0.5:
                t = nn.functional.pad(t, (4,4,4,4), mode='reflect')
                t = nn.functional.interpolate(t, size=(224,224), mode='bilinear')

            t = self.apply_maskout(t)

        if t.shape[-1] != 224:
            t = nn.functional.interpolate(t, size=(224,224))

        return t.to(self.device)


    # label 7
    def train(self, dataset, epochs=30, batch_size=32, do_plots=False):
        print("Loading dataset")

        images_pool = list(dataset.images_seq(dataset.n_files))
        gt_labels = np.array(dataset.labels)

        # label 7.1
        idx_train, idx_valid = train_test_split(
            np.arange(dataset.n_files),
            test_size=0.1,
            stratify=gt_labels,
            shuffle=True
        )

        logs = {"train_loss":[], "val_loss":[], "train_acc":[], "val_acc":[], "lr":[]}

        best_val = float('inf')
        self.net.train()

        # label 7.2
        for ep in range(1, epochs + 1):

            np.random.shuffle(idx_train)
            loss_sum = 0
            corr_counter = 0

            # label 7.3
            for start in tqdm(range(0, len(idx_train), batch_size), desc=f"Epoch {ep}"):
                batch_ids = idx_train[start:start+batch_size]

                batch_x = torch.cat([self.prepare_tensor(images_pool[i], True) for i in batch_ids])
                batch_y = torch.tensor([gt_labels[i] for i in batch_ids], dtype=torch.long, device=self.device)

                logits = self.net(batch_x)
                loss = self.loss_fn(logits, batch_y)

                self.optimizer.zero_grad()
                loss.backward()
                self.optimizer.step()

                loss_sum += loss.item() * len(batch_ids)
                corr_counter += (logits.argmax(1) == batch_y).sum().item()

            # label 7.4
            tr_loss = loss_sum / len(idx_train)
            tr_acc  = corr_counter / len(idx_train)

            val_loss = 0
            val_ok = 0

            self.net.eval()
            with torch.no_grad():
                for i in idx_valid:
                    xb = self.prepare_tensor(images_pool[i], False)
                    yb = torch.tensor([gt_labels[i]], device=self.device)

                    pred = self.net(xb)
```

```
                val_loss += self.loss_fn(pred, yb).item()
                val_ok   += (pred.argmax(1) == yb).sum().item()

        val_loss /= len(idx_valid)
        val_acc = val_ok / len(idx_valid)
        lr = self.optimizer.param_groups[0]["lr"]

        print(f"[Epoch {ep}] TL={tr_loss:.4f}, TA={tr_acc:.3f}, VL={val_loss:.4f}, VA={val_acc:.3f}, LR={lr:.5f}")

        # label 7.5
        self.scheduler.step(val_loss)

        # label 7.6
        if val_loss < best_val:
            best_val = val_loss
            self.save("best")

        logs["train_loss"].append(tr_loss)
        logs["val_loss"].append(val_loss)
        logs["train_acc"].append(tr_acc)
        logs["val_acc"].append(val_acc)
        logs["lr"].append(lr)

        self.net.train()

    if do_plots:
        self._plot(logs)

    return logs


# label 8
def _plot(self, logs):
    ep = range(1, len(logs["train_loss"]) + 1)

    plt.figure(figsize=(12,4))

    plt.subplot(1,2,1)
    plt.plot(ep, logs["train_loss"])
    plt.plot(ep, logs["val_loss"])
    plt.grid(True)

    plt.subplot(1,2,2)
    plt.plot(ep, logs["train_acc"])
    plt.plot(ep, logs["val_acc"])
    plt.grid(True)

    plt.show()


# label 9
def test_on_image(self, img):
    self.net.eval()
    with torch.no_grad():
        x = self.prepare_tensor(img, False)
        pred = self.net(x)
        return pred.argmax(1).item()


# label 10
def test_on_dataset(self, dataset, limit=None):
    self.net.eval()

    total = dataset.n_files if limit is None else int(dataset.n_files * limit)

    preds = []
    correct = 0

    for i, img in tqdm(enumerate(dataset.images_seq(total)), total=total):
        p = self.test_on_image(img)
        preds.append(p)
        if p == dataset.labels[i]:
            correct += 1

    acc = correct / total
    print(f"Accuracy: {acc*100:.2f}%")
    return preds


# label 11
def evaluate_with_confusion_matrix(self, dataset, limit=None, class_names=None):
    preds = self.test_on_dataset(dataset, limit)
    true = dataset.labels[:len(preds)]
```

```
cm = confusion_matrix(true, preds)
print(classification_report(true, preds, target_names=class_names))

plt.figure(figsize=(7, 6))
plt.imshow(cm, cmap='Blues')
plt.colorbar()

ticks = np.arange(len(cm))
if class_names is None:
    class_names = [str(i) for i in ticks]

plt.xticks(ticks, class_names, rotation=45)
plt.yticks(ticks, class_names)

thr = cm.max() / 2
for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
    plt.text(
        j, i, cm[i, j],
        ha="center",
        color="white" if cm[i,j] > thr else "black"
    )

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

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_small')
d_test = Dataset('test_small')
```

```
Downloading...
From: https://drive.google.com/uc?export=download&confirm=pbef&id=1qd45xXfDwdZjktLFwQb-et-mAaFeCzOR
To: /content/train_small.npz
100%|██████████| 841M/841M [00:14<00:00, 56.8MB/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=1wbRsog0n7uGlHIPGLhyN-PMeT2kdQ2lI
To: /content/test_small.npz
100%|██████████| 211M/211M [00:04<00:00, 42.2MB/s]
Loading dataset test_small from npz.
Done. Dataset test_small consists of 1800 images.
```

```
model = Model()
if not EVALUATE_ONLY:
    model.train(d_train)
    model.save('best')
else:
    model.load('best')
```

```
Loading dataset
Epoch 1: 100%|██████████| 203/203 [00:28<00:00,  7.22it/s]
[Epoch 1] TL=1.2786, TA=0.634, VL=0.9934, VA=0.781, LR=0.00100
Model saved to /content/drive/MyDrive/best.pth
Epoch 2: 100%|██████████| 203/203 [00:28<00:00,  7.22it/s]
[Epoch 2] TL=0.9759, TA=0.785, VL=0.8595, VA=0.850, LR=0.00100
Model saved to /content/drive/MyDrive/best.pth
Epoch 3: 100%|██████████| 203/203 [00:27<00:00,  7.27it/s]
[Epoch 3] TL=0.8573, TA=0.843, VL=0.9565, VA=0.822, LR=0.00100
Epoch 4: 100%|██████████| 203/203 [00:27<00:00,  7.27it/s]
[Epoch 4] TL=0.8101, TA=0.865, VL=0.7642, VA=0.906, LR=0.00100
Model saved to /content/drive/MyDrive/best.pth
Epoch 5: 100%|██████████| 203/203 [00:27<00:00,  7.26it/s]
[Epoch 5] TL=0.7585, TA=0.894, VL=0.8239, VA=0.883, LR=0.00100
Epoch 6: 100%|██████████| 203/203 [00:28<00:00,  7.16it/s]
[Epoch 6] TL=0.7288, TA=0.907, VL=0.7276, VA=0.917, LR=0.00100
Model saved to /content/drive/MyDrive/best.pth
Epoch 7: 100%|██████████| 203/203 [00:27<00:00,  7.26it/s]
[Epoch 7] TL=0.7170, TA=0.908, VL=0.7228, VA=0.928, LR=0.00100
Model saved to /content/drive/MyDrive/best.pth
Epoch 8: 100%|██████████| 203/203 [00:28<00:00,  7.16it/s]
[Epoch 8] TL=0.7031, TA=0.914, VL=0.7429, VA=0.918, LR=0.00100
Epoch 9: 100%|██████████| 203/203 [00:27<00:00,  7.29it/s]
[Epoch 9] TL=0.6891, TA=0.920, VL=1.5834, VA=0.717, LR=0.00100
Epoch 10: 100%|██████████| 203/203 [00:27<00:00,  7.27it/s]
```

```
[Epoch 10] TL=0.6775, TA=0.927, VL=0.6729, VA=0.939, LR=0.00100
Model saved to /content/drive/MyDrive/best.pth
Epoch 11: 100%|██████████| 203/203 [00:28<00:00,  7.23it/s]
[Epoch 11] TL=0.6814, TA=0.923, VL=0.6905, VA=0.936, LR=0.00100
Epoch 12: 100%|██████████| 203/203 [00:27<00:00,  7.25it/s]
[Epoch 12] TL=0.6707, TA=0.929, VL=0.6683, VA=0.942, LR=0.00100
Model saved to /content/drive/MyDrive/best.pth
Epoch 13: 100%|██████████| 203/203 [00:28<00:00,  7.21it/s]
[Epoch 13] TL=0.6584, TA=0.934, VL=0.7082, VA=0.921, LR=0.00100
Epoch 14: 100%|██████████| 203/203 [00:28<00:00,  7.22it/s]
[Epoch 14] TL=0.6431, TA=0.940, VL=0.6578, VA=0.946, LR=0.00100
Model saved to /content/drive/MyDrive/best.pth
Epoch 15: 100%|██████████| 203/203 [00:28<00:00,  7.09it/s]
[Epoch 15] TL=0.6448, TA=0.942, VL=0.7138, VA=0.914, LR=0.00100
Epoch 16: 100%|██████████| 203/203 [00:28<00:00,  7.11it/s]
[Epoch 16] TL=0.6278, TA=0.948, VL=0.6438, VA=0.949, LR=0.00100
Model saved to /content/drive/MyDrive/best.pth
Epoch 17: 100%|██████████| 203/203 [00:28<00:00,  7.18it/s]
[Epoch 17] TL=0.6378, TA=0.944, VL=0.6463, VA=0.944, LR=0.00100
Epoch 18: 100%|██████████| 203/203 [00:28<00:00,  7.09it/s]
[Epoch 18] TL=0.6327, TA=0.946, VL=0.6693, VA=0.939, LR=0.00100
Epoch 19: 100%|██████████| 203/203 [00:28<00:00,  7.07it/s]
[Epoch 19] TL=0.6238, TA=0.949, VL=0.6579, VA=0.939, LR=0.00100
Epoch 20: 100%|██████████| 203/203 [00:28<00:00,  7.14it/s]
[Epoch 20] TL=0.5985, TA=0.958, VL=0.6181, VA=0.958, LR=0.00050
Model saved to /content/drive/MyDrive/best.pth
Epoch 21: 100%|██████████| 203/203 [00:28<00:00,  7.10it/s]
[Epoch 21] TL=0.5915, TA=0.961, VL=0.6227, VA=0.961, LR=0.00050
Epoch 22: 100%|██████████| 203/203 [00:28<00:00,  7.04it/s]
[Epoch 22] TL=0.5853, TA=0.965, VL=0.6272, VA=0.963, LR=0.00050
Epoch 23: 100%|██████████| 203/203 [00:28<00:00,  7.12it/s]
[Epoch 23] TL=0.5858, TA=0.966, VL=0.6052, VA=0.967, LR=0.00050
```

Пример тестирования модели на части набора данных:

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

```
100%|██████████| 180/180 [00:00<00:00, 516.46it/s]Accuracy: 100.00%
metrics for 10% of test:
         accuracy 1.0000:
         balanced accuracy 1.0000:

/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:407: UserWarning: A single label was found i
  warnings.warn(
```

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%|██████████| 1800/1800 [00:03<00:00, 535.03it/s]Accuracy: 98.00%
metrics for test:
         accuracy 0.9800:
         balanced accuracy 0.9800:
```

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

Настоятельно рекомендуется после получения пайплайна с полными результатами обучения экспортировать ноутбук в pdf (файл -> печать) и прислать этот pdf вместе с самим ноутбуком.

Тестирование модели на других наборах данных

Ваша модель должна поддерживать тестирование на других наборах данных. Для удобства, Вам предоставляется набор данных test_tiny, который представляет собой малую часть (2% изображений) набора test. Ниже приведен фрагмент кода, который будет осуществлять тестирование для оценивания Вашей модели на дополнительных тестовых наборах данных.

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

```
final_model = Model()
final_model.load('best')
d_test_tiny = Dataset('test_tiny')
pred = final_model.test_on_dataset(d_test_tiny)
Metrics.print_all(d_test_tiny.labels, pred, 'test-tiny')
```

```
Model loaded from /content/drive/MyDrive/best.pth
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, 34.5MB/s]
Loading dataset test_tiny from npz.
Done. Dataset test_tiny consists of 90 images.
100%|██████████| 90/90 [00:00<00:00, 431.33it/s]Accuracy: 96.67%
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:

```
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.')
```

```
Function f is caluclated 128 times in 0.04164341199975752s.
```

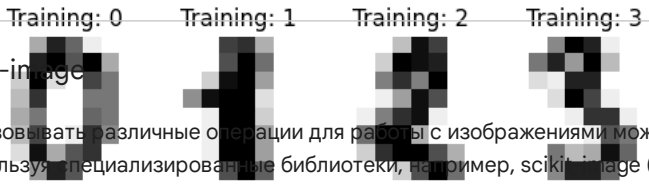
Scikit-learn

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)
```

```
_, 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()
```

```
Classification report for classifier SVC(gamma=0.001):
              precision    recall  f1-score   support

           0       1.00      0.99      0.99        88
           1       0.99      0.97      0.98        91
           2       0.99      0.99      0.99        86
           3       0.98      0.87      0.92        91
           4       0.99      0.96      0.97        92
           5       0.95      0.97      0.96        91
           6       0.99      0.99      0.99        91
           7       0.96      0.99      0.97        89
           8       0.94      1.00      0.97        88
           9       0.93      0.98      0.95        92

    accuracy                           0.97       899
   macro avg       0.97      0.97      0.97       899
weighted avg       0.97      0.97      0.97       899
```



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 нейронных моделей можно использовать фреймворк глубокого обучения Tensorflow 2. Ниже приведен пример простейшей нейронной сети, используемой 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 осущетвляться 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"
```