

✓ Задание по курсу "Практическое введение в анализ изображений"

Подготовила студентка группы 208 Пескова Карина

Для начала установим пакет petroscope.

```
!pip install petroscope --force-reinstall
```

↗ Показать скрытые выходные данные

Код ниже позволит работать с данными на диске

Путь до файлов на диске:

```
dev -> intro_course_prak -> LumenStone -> S1_v1.5
```

```
from pathlib import Path
from google.colab import drive

drive.mount('/content/drive')
ds_path = Path('/content/drive/MyDrive/dev/intro_course_prak/LumenStone/S1_v1')
```

↗ Mounted at /content/drive

Для работы с данными используем библиотеку petroscope, выведем номер, название и цвет класса каждого минерала.

```
from petroscope.segmentation.classes import ClassSet, LumenStoneClasses

classset = LumenStoneClasses.S1v1()
for cl in classset.classes:
    print(cl)
```

↗

```
[0, bg (background), color: #000000]
[1, ccp/kub (chalcopryrite/cubanite), color: #ffa500]
[2, gl (galena), color: #9acd32]
[4, brt (bornite), color: #00bfff]
[6, py/mrc (pyrite/marcasite), color: #2f4f4f]
[8, sph (sphalerite), color: #ee82ee]
[11, tnt/ttr (tenantite/tetrahedrite), color: #483d8b]
```

Для наглядности покажем прямоугольнички в цвет каждого минерала.

```
def show_class_color(cl):

    fig, ax = plt.subplots()

    e_color = "black"
    f_color = cl.color
    rect = patches.Rectangle((0.2, 0.2), 0.5, 0.5, facecolor=f_color)

    # Добавляем прямоугольник на оси
    ax.add_patch(rect)

    # Устанавливаем пределы осей для отображения прямоугольника
    ax.set_xlim(0, 1)
    ax.set_ylim(0, 1)

    title = cl.name
    # Отображаем график
    plt.title(title)
    plt.axis("off")
    plt.show()

from petroscope.segmentation.classes import ClassSet, LumenStoneClasses
import matplotlib.patches as patches
import matplotlib.pyplot as plt

classset = LumenStoneClasses.S1v1()
for cl in classset.classes:
    print("Class code: ", cl.code)
    show_class_color(cl)
```

 Class code: 0

background



Class code: 1

chalcopryite/cubanite



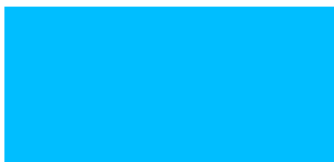
Class code: 2

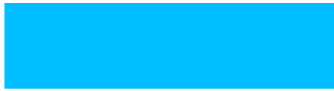
galena



Class code: 4

bornite





Class code: 6

pyrite/marcasite



Class code: 8

sphalerite



Class code: 11

tenantite/tetrahedrite



Оформим наш стандартный код для отображения изображений в функцию.

```
import numpy as np
import matplotlib.pyplot as plt

def show(image, title: str = None, cmap: str = None):
    plt.imshow(image, cmap=cmap)
    if title:
        plt.title(title)
    plt.axis('off')
    plt.show()
```

Переходим к написанию кода, который будет решать нашу задачу!

Импортируем все нужные библиотеки

```
from pathlib import Path
import numpy as np

import torch
import torch.nn as nn
import torch.optim as optim
import torchvision
import torchvision.transforms as T
from torch.utils.data import Dataset, DataLoader
from tqdm import tqdm
import torch.nn.functional as F

from petroscope.segmentation.classes import ClassSet, LumenStoneClasses
from petroscope.segmentation.utils import load_image, load_mask

from PIL import Image
```

Создадим класс нашего датасета

```
from PIL import Image
class SegmentationDataset(Dataset):
    def __init__(self, img_mask_paths, classes, transform_img=None, transform_mask=None):
        self.img_mask_paths = img_mask_paths
        self.classes = classes
        self.transform_img = transform_img
        self.transform_mask = transform_mask

    def __len__(self):
        return len(self.img_mask_paths)

    def __getitem__(self, idx):
        img_path, mask_path = self.img_mask_paths[idx]
        # загрузка изображения и маски
        img = load_image(img_path, normalize=True)
        mask = load_mask(mask_path, classes=self.classes, one_hot=False)

        # преобразование в Image
        img = Image.fromarray((img * 255).astype(np.uint8))
        mask = Image.fromarray(mask.astype(np.uint8))

        if self.transform_img:
            img = self.transform_img(img)
        if self.transform_mask:
            mask = self.transform_mask(mask)
            # приводим к long, тк будем использовать CrossEntropyLoss
            # в качестве функции потерь, а она работает только с целыми метками
            mask = mask.squeeze(0).long()
        return img, mask
```

Пояснения к коду ниже:

- Подготовим изображения для подачи в модель. Так как они очень большие, было принято решение их ресайзнуть: сами изображения аншлифов ресайзятся с помощью билинейной интерполяцией, а маски - методом К ближайших соседей. Также изображения мы приводим к формату тензора для того, чтобы передать их модели
- В качестве модели была выбрана модель resnet50. Изначально эта модель рассчитана на 21 класс, а у нас их 7 (6 минералов + фон), поэтому последний сверточный слой был заменен
- В качестве оптимизатора используется Adam (по советам Александра Владимировича :))

- В качестве функции потерь - CrossEntropyLoss

```
import petroscope.segmentation as segm

class SegmModel(segm.GeoSegmModel):
    def __init__(self, classes):
        super().__init__()
        self.classes = classes
        self.num_classes = len(classes.classes)
        self.device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

        # ресайзим изображения и маски
        self.transform_img = T.Compose([
            T.Resize((512, 512), interpolation=T.InterpolationMode.BILINEAR),
            T.ToTensor()
        ])
        self.transform_mask = T.Compose([
            T.Resize((512, 512), interpolation=T.InterpolationMode.NEAREST),
            T.PILToTensor()
        ])

        # модель
        self.model = torchvision.models.segmentation.fcn_resnet50(pretrained=True)
        in_channels = self.model.classifier[4].in_channels
        self.model.classifier[4] = nn.Conv2d(in_channels, self.num_classes, kernel_size=1)
        self.model = self.model.to(self.device)

    def train(self, img_mask_paths, epochs=5, batch_size=4, lr=1e-4, **kwargs):
        dataset = SegmentationDataset(
            img_mask_paths, self.classes,
            transform_img=self.transform_img,
            transform_mask=self.transform_mask
        )
        dataloader = DataLoader(dataset, batch_size=batch_size, shuffle=True)

        optimizer = torch.optim.Adam(self.model.parameters(), lr=lr)
        criterion = nn.CrossEntropyLoss()

        for epoch in range(epochs):
            self.model.train()
            if epoch == 0:
                print(f"step {epoch+1}/{epochs} loss: ---")
            else:
                print(f"step {epoch+1}/{epochs} loss: {total_loss/len(dataloader):.4f}")

            total_loss = 0
            for images, masks in tqdm(dataloader):
                images, masks = images.to(self.device), masks.to(self.device)
                optimizer.zero_grad()
                outputs = self.model(images)['out']
                loss = criterion(outputs, masks)
                loss.backward()
                optimizer.step()
                total_loss += loss.item()

    def predict_image(self, image: np.ndarray) -> np.ndarray:
        self.model.eval()
        with torch.no_grad():
            s = image.shape[:2]
            image_pil = Image.fromarray((image * 255).astype(np.uint8))
            input_tensor = self.transform_img(image_pil).unsqueeze(0).to(self.device)

            output = self.model(input_tensor)['out']
            # возвращаем оригинальные размеры
            output = F.interpolate(output, size=s, mode='bilinear', align_corners=False)
            prediction = torch.argmax(output.squeeze(), dim=0).cpu().numpy()
        return prediction

    def load(self, saved_path: Path, **kwargs):
        self.model.load_state_dict(torch.load(saved_path, map_location=self.device))

    def save(self, save_path: Path, **kwargs):
        torch.save(self.model.state_dict(), save_path)
```

Сформируем списки путей обучающих и тестовых изображений и обучим нашу модель

```
ds_path = Path('/content/drive/MyDrive/dev/intro_course_prak/LumenStone/S1_v1')
```

```
train_img_mask_p = [
```

```

(img_p, ds_path / "masks" / "train" / f"{img_p.stem}.png")
for img_p in sorted((ds_path / "imgs" / "train").iterdir())
]
test_img_mask_p = [
    (img_p, ds_path / "masks" / "test" / f"{img_p.stem}.png")
    for img_p in sorted((ds_path / "imgs" / "test").iterdir())
]

classset = LumenStoneClasses.S1v1()
model = SegmModel(classset)
model.train(train_img_mask_p, epochs=30)

```

```

↩ step 2/30 loss: 1.4192
100% ██████████ | 15/15 [00:44<00:00, 2.98s/it]
step 3/30 loss: 0.9638
100% ██████████ | 15/15 [00:44<00:00, 2.94s/it]
step 4/30 loss: 0.8692
100% ██████████ | 15/15 [00:44<00:00, 2.98s/it]
step 5/30 loss: 0.7556
100% ██████████ | 15/15 [00:44<00:00, 2.95s/it]
step 6/30 loss: 0.6094
100% ██████████ | 15/15 [00:44<00:00, 2.99s/it]
step 7/30 loss: 0.5477
100% ██████████ | 15/15 [00:45<00:00, 3.01s/it]
step 8/30 loss: 0.5469
100% ██████████ | 15/15 [00:44<00:00, 2.95s/it]
step 9/30 loss: 0.4693
100% ██████████ | 15/15 [00:44<00:00, 2.97s/it]
step 10/30 loss: 0.4246
100% ██████████ | 15/15 [00:44<00:00, 2.98s/it]
step 11/30 loss: 0.3853
100% ██████████ | 15/15 [00:45<00:00, 3.01s/it]
step 12/30 loss: 0.3721
100% ██████████ | 15/15 [00:45<00:00, 3.00s/it]
step 13/30 loss: 0.3431
100% ██████████ | 15/15 [00:44<00:00, 2.95s/it]
step 14/30 loss: 0.3254
100% ██████████ | 15/15 [00:44<00:00, 2.96s/it]
step 15/30 loss: 0.3165
100% ██████████ | 15/15 [00:44<00:00, 2.98s/it]
step 16/30 loss: 0.3538
100% ██████████ | 15/15 [00:44<00:00, 2.99s/it]
step 17/30 loss: 0.3032
100% ██████████ | 15/15 [00:44<00:00, 2.98s/it]
step 18/30 loss: 0.2976
100% ██████████ | 15/15 [00:44<00:00, 2.94s/it]
step 19/30 loss: 0.3193
100% ██████████ | 15/15 [00:44<00:00, 2.96s/it]
step 20/30 loss: 0.2695
100% ██████████ | 15/15 [00:44<00:00, 2.96s/it]
step 21/30 loss: 0.2528
100% ██████████ | 15/15 [00:44<00:00, 2.98s/it]
step 22/30 loss: 0.2726
100% ██████████ | 15/15 [00:44<00:00, 2.99s/it]
step 23/30 loss: 0.2790
100% ██████████ | 15/15 [00:44<00:00, 2.95s/it]
step 24/30 loss: 0.2256
100% ██████████ | 15/15 [00:45<00:00, 3.00s/it]
step 25/30 loss: 0.2284
100% ██████████ | 15/15 [00:44<00:00, 2.97s/it]
step 26/30 loss: 0.2254
100% ██████████ | 15/15 [00:44<00:00, 2.96s/it]
step 27/30 loss: 0.2184
100% ██████████ | 15/15 [00:44<00:00, 2.99s/it]
step 28/30 loss: 0.2234
100% ██████████ | 15/15 [00:44<00:00, 2.94s/it]
step 29/30 loss: 0.2083
100% ██████████ | 15/15 [00:44<00:00, 2.98s/it]
step 30/30 loss: 0.2147
100% ██████████ | 15/15 [00:44<00:00, 2.99s/it]

```

```

# сохраним веса модели
save_path = Path("weights.pth")
model.save(save_path)

```

Пришло время для тестирования модели :)

```

from petroscope.segmentation.eval import SegmDetailedTester

num_classes = 7

load_model = SegmModel(classset)
load_model.load(save_path)

tester = SegmDetailedTester(

```

```

    Path("output"),
    classes=classset,
    void_pad=0,
    void_border_width=4,
    vis_plots=False,
    vis_segmentation=True,
)

res, res_void = tester.test_on_set(
    test_img_mask_p,
    lambda img: load_model.predict_image(img),
    description="Results",
    return_void=True,
)

print(f"Metrics:\n{res}")
print(f"Metrics with void borders:\n{res_void}")

↩ testing: 100%|██████████| 16/16 [02:54<00:00, 10.90s/it]Metrics:
    iou [soft]:
        bg: 0.7839 [0.7839]
        brt: 0.8649 [0.8649]
        ccp/kub: 0.7406 [0.7406]
        gl: 0.6490 [0.6490]
        py/mrc: 0.8983 [0.8983]
        sph: 0.7397 [0.7397]
        tnt/ttr: 0.7516 [0.7516]
    mean iou [soft]: 0.7754 [0.7754]
    acc: 0.8910

Metrics with void borders:
    iou [soft]:
        bg: 0.8308 [0.8308]
        brt: 0.8900 [0.8900]
        ccp/kub: 0.7794 [0.7794]
        gl: 0.6880 [0.6880]
        py/mrc: 0.9296 [0.9296]
        sph: 0.7794 [0.7794]
        tnt/ttr: 0.7856 [0.7856]
    mean iou [soft]: 0.8118 [0.8118]
    acc: 0.9198

```

Визуализируем наши результаты. После тестирования они хранятся в папке Results.

```

from PIL import Image

for i in range(1, 17):

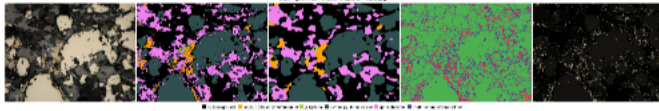
    num = str(i)
    if i <= 9:
        num = "0" + num

    img_path = f"/content/output/Results/img_{num}_composite.jpg"
    img = Image.open(img_path)
    show(img, f"Picture number {i}")

```



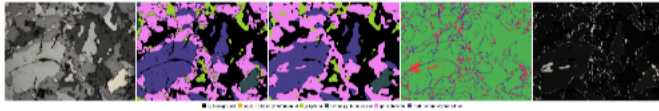
Picture number 1



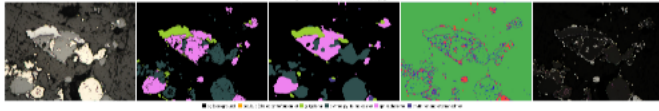
Picture number 2



Picture number 3



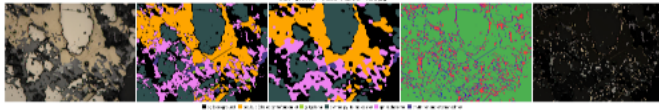
Picture number 4



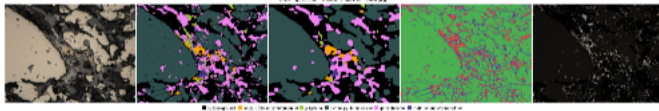
Picture number 5



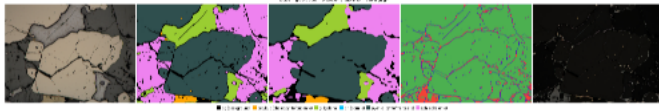
Picture number 6



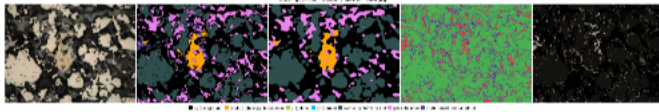
Picture number 7



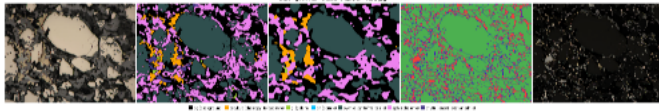
Picture number 8



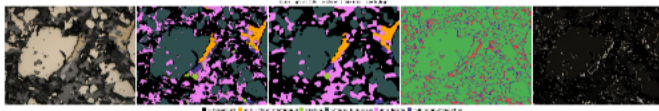
Picture number 9



Picture number 10



Picture number 11



Picture number 12



Picture number 13

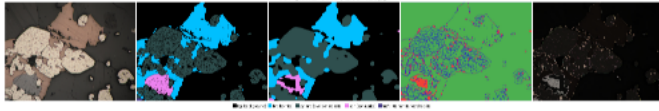




Picture number 14



Picture number 15



Picture number 16

