Чтобы изменить содержимое ячейки, дважды нажмите на нее (или выберите "Ввод")

```
Подключение google disk для использовани
```

Подключение google disk для использования dataset

```
from google.colab import drive
drive.mount('/content/drive')
```

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

Установка библиотеки

Чтобы изменить содержимое ячейки, дважды нажмите на нее (или выберите "Ввод")

```
!pip install petroscope
```

```
Requirement already satisfied: petroscope in /usr/local/lib/python3.11/dist
Requirement already satisfied: pyyaml in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: numpy<2.0.0,>=1.16 in /usr/local/lib/python3
Requirement already satisfied: pillow in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: scipy in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: loguru in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: prettytable in /usr/local/lib/python3.11/dis
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.1
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/di
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.1
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/pytho
Requirement already satisfied: wcwidth in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/p
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/di
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-p
```

Импорт неободимых библиотек

```
from pathlib import Path
from petroscope.segmentation.classes import ClassSet, LumenStoneClasses
from petroscope.segmentation.utils import load_image, load_mask
from petroscope.segmentation import GeoSegmModel
import numpy as np
from tqdm import tqdm
import tensorflow as tf
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D, UpSampling2D,
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical
from skimage.color import rgb2lab
import os
```

Путь к dataset

```
ds_path = Path('/content/drive/MyDrive/PhotoSets/')
```

Инциализация набора классов для сегментации

```
classset = LumenStoneClasses.S1v1()

for cl in classset.classes:
    print(cl)
```

```
[0, bg (background), color: #000000]
[1, ccp/kub (chalcopyrite/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]
```

Формирование путей к изображениям и маскам

```
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())
]
```

Загрузка и конвертация обучающих изображений в пространство LAB

```
for img_p, _ in train_img_mask_p:
    img = load_image(img_p, normalize=True)
    img_lab = rgb2lab(img)
    print(f"Image {img_p.name}: {img_lab.shape}, {img_lab.dtype}")
```

```
Image train_01.jpg: (2547, 3396, 3), float32
Image train_02.jpg: (2547, 3396, 3), float32
Image train_03.jpg: (2547, 3396, 3), float32
Image train_04.jpg: (2547, 3396, 3), float32
Image train_06.jpg: (2547, 3396, 3), float32
Image train_07.jpg: (2547, 3396, 3), float32
Image train_08.jpg: (2547, 3396, 3), float32
Image train_09.jpg: (2547, 3396, 3), float32
Image train_10.jpg: (2547, 3396, 3), float32
Image train_11.jpg: (2547, 3396, 3), float32
Image train_12.jpg: (2547, 3396, 3), float32
Image train_13.jpg: (2547, 3396, 3), float32
Image train_14.jpg: (2547, 3396, 3), float32
Image train_15.jpg: (2547, 3396, 3), float32
Image train_16.jpg: (2547, 3396, 3), float32
Image train_17.jpg: (2547, 3396, 3), float32
Image train_18.jpg: (2547, 3396, 3), float32
Image train_19.jpg: (2547, 3396, 3), float32
Image train_20.jpg: (2547, 3396, 3), float32
Image train_21.jpg: (2547, 3396, 3), float32
Image train_22.jpg: (2547, 3396, 3), float32
Image train_23.jpg: (2547, 3396, 3), float32
Image train_24.jpg: (2547, 3396, 3), float32
Image train_25.jpg: (2547, 3396, 3), float32
Image train_26.jpg: (2547, 3396, 3), float32
Image train_27.jpg: (2547, 3396, 3), float32
Image train_28.jpg: (2547, 3396, 3), float32
Image train_29.jpg: (2547, 3396, 3), float32
Image train_30.jpg: (2547, 3396, 3), float32
Image train_31.jpg: (2547, 3396, 3), float32
Image train_32.jpg: (2547, 3396, 3), float32
Image train_33.jpg: (2547, 3396, 3), float32
Image train_34.jpg: (2547, 3396, 3), float32
Image train_35.jpg: (2547, 3396, 3), float32
Image train_36.jpg: (2547, 3396, 3), float32
Image train_37.jpg: (2547, 3396, 3), float32
Image train_38.jpg: (2547, 3396, 3), float32
Image train_39.jpg: (2547, 3396, 3), float32
Image train_40.jpg: (2547, 3396, 3), float32
Image train_41.jpg: (2547, 3396, 3), float32
Image train_42.jpg: (2547, 3396, 3), float32
Image train_43.jpg: (2547, 3396, 3), float32
Image train_44.jpg: (2547, 3396, 3), float32
Image train_45.jpg: (2547, 3396, 3), float32
Image train_46.jpg: (2547, 3396, 3), float32
Image train_47.jpg: (2547, 3396, 3), float32
Image train_48.jpg: (2547, 3396, 3), float32
Image train_49.jpg: (2547, 3396, 3), float32
Image train_50.jpg: (2547, 3396, 3), float32
Image train_51.jpg: (2547, 3396, 3), float32
Image train_52.jpg: (2547, 3396, 3), float32
Image train_53.jpg: (2547, 3396, 3), float32
Image train_54.jpg: (2547, 3396, 3), float32
Image train_55.jpg: (2547, 3396, 3), float32
Image train_56.jpg: (2547, 3396, 3), float32
Image train_57.jpg: (2547, 3396, 3), float32
Image train_58.jpg: (2547, 3396, 3), float32
Image train_59.jpg: (2547, 3396, 3), float32
```

Загрузка масок без one-hot кодирования

```
for _, mask_p in train_img_mask_p:
    mask = load_mask(mask_p, classes=classset, one_hot=False)
    print(f"Mask {mask_p.name}: {mask.shape}, {mask.dtype}")
```

```
Mask train_01.png: (2547, 3396), uint8
Mask train_02.png: (2547, 3396), uint8
Mask train_03.png: (2547, 3396), uint8
Mask train_04.png: (2547, 3396), uint8
Mask train_06.png: (2547, 3396), uint8
Mask train_07.png: (2547, 3396), uint8
Mask train_08.png: (2547, 3396), uint8
Mask train_09.png: (2547, 3396), uint8
Mask train_10.png: (2547, 3396), uint8
Mask train_11.png: (2547, 3396), uint8
Mask train_12.png: (2547, 3396), uint8
Mask train_13.png: (2547, 3396), uint8
Mask train_14.png: (2547, 3396), uint8
Mask train_15.png: (2547, 3396), uint8
Mask train_16.png: (2547, 3396), uint8
Mask train_17.png: (2547, 3396), uint8
Mask train_18.png: (2547, 3396), uint8
Mask train_19.png: (2547, 3396), uint8
Mask train_20.png: (2547, 3396), uint8
Mask train_21.png: (2547, 3396), uint8
Mask train_22.png: (2547, 3396), uint8
Mask train_23.png: (2547, 3396), uint8
Mask train_24.png: (2547, 3396), uint8
Mask train_25.png: (2547, 3396), uint8
Mask train_26.png: (2547, 3396), uint8
Mask train_27.png: (2547, 3396), uint8
Mask train_28.png: (2547, 3396), uint8
Mask train_29.png: (2547, 3396), uint8
Mask train_30.png: (2547, 3396), uint8
Mask train_31.png: (2547, 3396), uint8
Mask train_32.png: (2547, 3396), uint8
Mask train_33.png: (2547, 3396), uint8
Mask train_34.png: (2547, 3396), uint8
Mask train_35.png: (2547, 3396), uint8
Mask train_36.png: (2547, 3396), uint8
Mask train_37.png: (2547, 3396), uint8
Mask train_38.png: (2547, 3396), uint8
Mask train_39.png: (2547, 3396), uint8
Mask train_40.png: (2547, 3396), uint8
Mask train_41.png: (2547, 3396), uint8
Mask train_42.png: (2547, 3396), uint8
Mask train_43.png: (2547, 3396), uint8
Mask train_44.png: (2547, 3396), uint8
Mask train_45.png: (2547, 3396), uint8
Mask train_46.png: (2547, 3396), uint8
Mask train_47.png: (2547, 3396), uint8
Mask train_48.png: (2547, 3396), uint8
Mask train_49.png: (2547, 3396), uint8
Mask train_50.png: (2547, 3396), uint8
Mask train_51.png: (2547, 3396), uint8
Mask train_52.png: (2547, 3396), uint8
Mask train_53.png: (2547, 3396), uint8
Mask train_54.png: (2547, 3396), uint8
Mask train_55.png: (2547, 3396), uint8
Mask train_56.png: (2547, 3396), uint8
Mask train_57.png: (2547, 3396), uint8
Mask train_58.png: (2547, 3396), uint8
Mask train_59.png: (2547, 3396), uint8
```

Загрузка масок с one-hot кодированием

```
for _, mask_p in train_img_mask_p:
    mask_one_hot = load_mask(mask_p, classes=classset, one_hot=True)
    print(f"Mask one-hot {mask_p.name}: {mask_one_hot.shape}, {mask_one_hot.dty
```

```
Mask one-hot train_01.png: (2547, 3396, 7), float32
Mask one-hot train_02.png: (2547, 3396, 7), float32
Mask one-hot train_03.png: (2547, 3396, 7), float32
Mask one-hot train_04.png: (2547, 3396, 7), float32
Mask one-hot train_06.png: (2547, 3396, 7), float32
Mask one-hot train_06.png: (2547, 3396, 7), float32
Mask one-hot train_07.png: (2547, 3396, 7), float32
Mask one-hot train_08.png: (2547, 3396, 7), float32
Mask one-hot train_09.png: (2547, 3396, 7), float32
Mask one-hot train_10.png: (2547, 3396, 7), float32
Mask one-hot train_11.png: (2547, 3396, 7), float32
Mask one-hot train_12.png: (2547, 3396, 7), float32
Mask one-hot train_13.png: (2547, 3396, 7), float32
Mask one-hot train_14.png: (2547, 3396, 7), float32
Mask one-hot train_15.png: (2547, 3396, 7), float32
Mask one-hot train_16.png: (2547, 3396, 7), float32
Mask one-hot train_17.png: (2547, 3396, 7), float32
Mask one-hot train_18.png: (2547, 3396, 7), float32
Mask one-hot train_19.png: (2547, 3396, 7), float32
Mask one-hot train_20.png: (2547, 3396, 7), float32
Mask one-hot train_21.png: (2547, 3396, 7), float32
Mask one-hot train_22.png: (2547, 3396, 7), float32
Mask one-hot train_23.png: (2547, 3396, 7), float32
Mask one-hot train_24.png: (2547, 3396, 7), float32
Mask one-hot train_25.png: (2547, 3396, 7), float32
Mask one-hot train_26.png: (2547, 3396, 7), float32
Mask one-hot train_27.png: (2547, 3396, 7), float32
Mask one-hot train_28.png: (2547, 3396, 7), float32
Mask one-hot train_29.png: (2547, 3396, 7), float32
Mask one-hot train_30.png: (2547, 3396, 7), float32
Mask one-hot train_31.png: (2547, 3396, 7), float32
Mask one-hot train_32.png: (2547, 3396, 7), float32
Mask one-hot train_33.png: (2547, 3396, 7), float32
Mask one-hot train_34.png: (2547, 3396, 7), float32
Mask one-hot train_35.png: (2547, 3396, 7), float32
Mask one-hot train_36.png: (2547, 3396, 7), float32
Mask one-hot train_37.png: (2547, 3396, 7), float32
Mask one-hot train_38.png: (2547, 3396, 7), float32
Mask one-hot train_39.png: (2547, 3396, 7), float32
Mask one-hot train_40.png: (2547, 3396, 7), float32
Mask one-hot train_41.png: (2547, 3396, 7), float32
Mask one-hot train_42.png: (2547, 3396, 7), float32
Mask one-hot train_43.png: (2547, 3396, 7), float32
Mask one-hot train_44.png: (2547, 3396, 7), float32
Mask one-hot train_45.png: (2547, 3396, 7), float32
Mask one-hot train_46.png: (2547, 3396, 7), float32
Mask one-hot train_47.png: (2547, 3396, 7), float32
Mask one-hot train_48.png: (2547, 3396, 7), float32
Mask one-hot train_49.png: (2547, 3396, 7), float32
Mask one-hot train_50.png: (2547, 3396, 7), float32
Mask one-hot train_51.png: (2547, 3396, 7), float32
Mask one-hot train_52.png: (2547, 3396, 7), float32
Mask one-hot train_53.png: (2547, 3396, 7), float32
Mask one-hot train_54.png: (2547, 3396, 7), float32

Mask one-hot train_55.png: (2547, 3396, 7), float32
Mask one-hot train_56.png: (2547, 3396, 7), float32
Mask one-hot train_57.png: (2547, 3396, 7), float32
Mask one-hot train_58.png: (2547, 3396, 7), float32
Mask one-hot train_59.png: (2547, 3396, 7), float32
```

Загрузка цветных масок

```
for img_p, _ in train_img_mask_p:
    mask_colored_path = ds_path / "masks_colored_png" / "train" / f"{img_p.sten
    mask_colored = load_image(mask_colored_path, normalize=False)
    print(f"Colored mask {mask_colored_path.name}: {mask_colored.shape}, {mask_
```

```
Colored mask train_01.png: (2547, 3396, 3), uint8
Colored mask train_02.png: (2547, 3396, 3), uint8
Colored mask train_03.png: (2547, 3396, 3), uint8
Colored mask train_04.png: (2547, 3396, 3), uint8
Colored mask train_06.png: (2547, 3396, 3), uint8
Colored mask train_07.png: (2547, 3396, 3), uint8
Colored mask train_08.png: (2547, 3396, 3), uint8
Colored mask train_09.png: (2547, 3396, 3), uint8
Colored mask train_10.png: (2547, 3396, 3), uint8
Colored mask train_11.png: (2547, 3396, 3), uint8
Colored mask train_12.png: (2547, 3396, 3), uint8
Colored mask train_13.png: (2547, 3396, 3), uint8
Colored mask train_14.png: (2547, 3396, 3), uint8
Colored mask train_15.png: (2547, 3396, 3), uint8
Colored mask train_16.png: (2547, 3396, 3), uint8
Colored mask train_17.png: (2547, 3396, 3), uint8
Colored mask train_18.png: (2547, 3396, 3), uint8
Colored mask train_19.png: (2547, 3396, 3), uint8
Colored mask train_20.png: (2547, 3396, 3), uint8
Colored mask train_21.png: (2547, 3396, 3), uint8
Colored mask train_22.png: (2547, 3396, 3), uint8
Colored mask train_23.png: (2547, 3396, 3), uint8
Colored mask train_24.png: (2547, 3396, 3), uint8
Colored mask train_25.png: (2547, 3396, 3), uint8
Colored mask train_26.png: (2547, 3396, 3), uint8
Colored mask train_27.png: (2547, 3396, 3), uint8
Colored mask train_28.png: (2547, 3396, 3), uint8
Colored mask train_29.png: (2547, 3396, 3), uint8
Colored mask train_30.png: (2547, 3396, 3), uint8
Colored mask train_31.png: (2547, 3396, 3), uint8
Colored mask train_32.png: (2547, 3396, 3), uint8
Colored mask train_33.png: (2547, 3396, 3), uint8
Colored mask train_34.png: (2547, 3396, 3), uint8
Colored mask train_35.png: (2547, 3396, 3), uint8
Colored mask train_36.png: (2547, 3396, 3), uint8
Colored mask train_37.png: (2547, 3396, 3), uint8
Colored mask train_38.png: (2547, 3396, 3), uint8
Colored mask train_39.png: (2547, 3396, 3), uint8
Colored mask train_40.png: (2547, 3396, 3), uint8
Colored mask train_41.png: (2547, 3396, 3), uint8
Colored mask train_42.png: (2547, 3396, 3), uint8
Colored mask train_43.png: (2547, 3396, 3), uint8
Colored mask train_44.png: (2547, 3396, 3), uint8
Colored mask train_45.png: (2547, 3396, 3), uint8
Colored mask train_46.png: (2547, 3396, 3), uint8
Colored mask train_47.png: (2547, 3396, 3), uint8
Colored mask train_48.png: (2547, 3396, 3), uint8
Colored mask train_49.png: (2547, 3396, 3), uint8
Colored mask train_50.png: (2547, 3396, 3), uint8
Colored mask train_51.png: (2547, 3396, 3), uint8
Colored mask train_52.png: (2547, 3396, 3), uint8
Colored mask train_53.png: (2547, 3396, 3), uint8
Colored mask train_54.png: (2547, 3396, 3), uint8
Colored mask train_55.png: (2547, 3396, 3), uint8
Colored mask train_56.png: (2547, 3396, 3), uint8
Colored mask train_57.png: (2547, 3396, 3), uint8
Colored mask train_58.png: (2547, 3396, 3), uint8
Colored mask train_59.png: (2547, 3396, 3), uint8
```

Функция для загрузки и предобработки изображений и масок

```
def load_and_preprocess(img_path, mask_path, classes, img_size=(256, 256)):
    img = load_image(img_path, normalize=True)
    img = tf.image.resize(img, img_size)

    mask = load_mask(mask_path, classes=classes, one_hot=False)
    mask = tf.image.resize(mask[..., np.newaxis], img_size, method='nearest')
    mask = to_categorical(mask, num_classes=len(classes))

    return img, mask
```

Унаследованный класс от GeoSegmModel, метод сегментации - нейронная сеть U-Net

```
class UNetSegmModel(GeoSegmModel):
    def __init__(self, classes: ClassSet, input_size=(256, 256, 3), save_path='
        super().__init__()
        self.classes = classes
        self.input_size = input_size
        self.save_path = Path(save_path)
        self.model = self._build_model()

    def _build_model(self):
        inputs = Input(self.input_size)

        conv1 = Conv2D(64, 3, activation='relu', padding='same')(inputs)
        conv1 = Conv2D(64, 3, activation='relu', padding='same')(conv1)
        pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)
        conv2 = Conv2D(128, 3, activation='relu', padding='same')(pool1)
        conv2 = Conv2D(128, 3, activation='relu', padding='same')(conv2)
        pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)
        conv3 = Conv2D(256, 3, activation='relu', padding='same')(pool2)
        conv3 = Conv2D(256, 3, activation='relu', padding='same')(conv3)
        pool3 = MaxPooling2D(pool_size=(2, 2))(conv3)
        conv4 = Conv2D(512, 3, activation='relu', padding='same')(pool3)
        conv4 = Conv2D(512, 3, activation='relu', padding='same')(conv4)

        up5 = concatenate([UpSampling2D(size=(2, 2))(conv4), conv3], axis=-1)
        conv5 = Conv2D(256, 3, activation='relu', padding='same')(up5)
        conv5 = Conv2D(256, 3, activation='relu', padding='same')(conv5)
        up6 = concatenate([UpSampling2D(size=(2, 2))(conv5), conv2], axis=-1)
        conv6 = Conv2D(128, 3, activation='relu', padding='same')(up6)
        conv6 = Conv2D(128, 3, activation='relu', padding='same')(conv6)
        up7 = concatenate([UpSampling2D(size=(2, 2))(conv6), conv1], axis=-1)
        conv7 = Conv2D(64, 3, activation='relu', padding='same')(up7)
        conv7 = Conv2D(64, 3, activation='relu', padding='same')(conv7)

        outputs = Conv2D(len(self.classes), 1, activation='softmax')(conv7)

        model = Model(inputs=[inputs], outputs=[outputs])
        model.compile(optimizer=Adam(), loss='categorical_crossentropy', metric
        return model

    def train(self, img_mask_paths, epochs=20, batch_size=4, validation_split=0
        train_data = [load_and_preprocess(img_p, mask_p, self.classes) for img_
        train_images, train_masks = zip(*train_data)
        train_images = np.array(train_images)
        train_masks = np.array(train_masks)

        self.model.fit([train_images], [train_masks], batch_size=batch_size, ep

        self.save()

    def predict_image(self, image: np.ndarray) -> np.ndarray:
        img_resized = tf.image.resize(image, self.input_size[:2])
        pred = self.model.predict(np.expand_dims(img_resized, axis=0))
        pred = np.argmax(pred[0], axis=-1)
        return pred.astype(np.uint8)

    def save(self):
        self.save_path.mkdir(parents=True, exist_ok=True)
        self.model.save(self.save_path / "unet_model.keras")

    def load(self, saved_path: Path, **kwargs):
        if not saved_path.exists():
            raise FileNotFoundError(f"Model file not found at {saved_path}")
        self.model = load_model(saved_path / "unet_model.keras")
```

Обучение модели

```
model = UNetSegmModel(classes=classset, input_size=(256, 256, 3), save_path="une

model.train(train_img_mask_p, epochs=200, batch_size=8, validation_split=0.1)
```

```
Loading data: 100%|██████████| 58/58 [00:32<00:00,  1.77it/s]
Epoch 1/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 81s 6s/step - accuracy: 0.2878 - loss: 3.1463 - va
Epoch 2/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 9s 335ms/step - accuracy: 0.3057 - loss: 1.7832 -
Epoch 3/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 2s 345ms/step - accuracy: 0.4016 - loss: 1.4435 -
Epoch 4/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 2s 337ms/step - accuracy: 0.5407 - loss: 1.2612 -
Epoch 5/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 3s 336ms/step - accuracy: 0.6613 - loss: 1.1003 -
Epoch 6/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 2s 336ms/step - accuracy: 0.6560 - loss: 1.0465 -
Epoch 7/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 3s 335ms/step - accuracy: 0.7034 - loss: 0.9920 -
Epoch 8/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 3s 350ms/step - accuracy: 0.7238 - loss: 0.9609 -
Epoch 9/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 2s 339ms/step - accuracy: 0.6060 - loss: 1.1678 -
Epoch 10/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 2s 337ms/step - accuracy: 0.7538 - loss: 0.8598 -
Epoch 11/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 3s 346ms/step - accuracy: 0.7084 - loss: 0.9119 -
Epoch 12/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 2s 339ms/step - accuracy: 0.7086 - loss: 0.9051 -
Epoch 13/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 2s 341ms/step - accuracy: 0.7061 - loss: 0.9036 -
Epoch 14/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 3s 349ms/step - accuracy: 0.7462 - loss: 0.8609 -
Epoch 15/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 2s 339ms/step - accuracy: 0.7326 - loss: 0.8531 -
Epoch 16/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 3s 339ms/step - accuracy: 0.8057 - loss: 0.7076 -
Epoch 17/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 3s 346ms/step - accuracy: 0.7050 - loss: 0.9211 -
Epoch 18/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 2s 351ms/step - accuracy: 0.7495 - loss: 0.7848 -
Epoch 19/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 3s 351ms/step - accuracy: 0.7498 - loss: 0.8005 -
Epoch 20/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 2s 339ms/step - accuracy: 0.7232 - loss: 0.9005 -
Epoch 21/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 3s 348ms/step - accuracy: 0.7883 - loss: 0.6885 -
Epoch 22/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 2s 341ms/step - accuracy: 0.8146 - loss: 0.6339 -
```

```
Epoch 23/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 2s 343ms/step - accuracy: 0.8132 - loss: 0.6357 -
Epoch 24/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 2s 345ms/step - accuracy: 0.7803 - loss: 0.6936 -
Epoch 25/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 3s 351ms/step - accuracy: 0.8023 - loss: 0.6354 -
Epoch 26/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 2s 349ms/step - accuracy: 0.8024 - loss: 0.6179 -
Epoch 27/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 3s 351ms/step - accuracy: 0.7927 - loss: 0.6709 -
Epoch 28/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 2s 351ms/step - accuracy: 0.7955 - loss: 0.6365 -
Epoch 29/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 3s 357ms/step - accuracy: 0.8024 - loss: 0.6401 -
```

Тестирование

```
from petroscope.segmentation.eval import SegmDetailedTester
from tensorflow.keras.utils import to_categorical

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

for img_p, mask_p in test_img_mask_p:
    img = load_image(img_p, normalize=True)

    pred = model.predict_image(img)
    print(f"Pred shape: {pred.shape}")

    mask = load_mask(mask_p, classes=classset, one_hot=False)
    mask_resized = tf.image.resize(mask[..., np.newaxis], (256, 256), method='r
    mask_resized = mask_resized[..., 0].numpy().astype(np.uint8)
    print(f"Mask resized shape: {mask_resized.shape}")

    pred_one_hot = to_categorical(pred, num_classes=len(classset.classes))
    mask_one_hot = to_categorical(mask_resized, num_classes=len(classset.classe
    print(f"Pred one-hot shape: {pred_one_hot.shape}")
    print(f"Mask one-hot shape: {mask_one_hot.shape}")

    metrics = tester.eval.evaluate(pred_one_hot, gt=mask_one_hot)
    metrics_void = tester.eval_void.evaluate(pred_one_hot, gt=mask_one_hot)

    print(f"Metrics for {img_p.name}:\n{metrics}")
    print(f"Metrics with void borders for {img_p.name}:\n{metrics_void}")
    print("-" * 50)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/models/functional.py:237:
Expected: ['keras_tensor']
Received: inputs=Tensor(shape=(1, 256, 256, 3))
  warnings.warn(msg)
1/1 ━━━━━━━━━━━━━━━━━━━━ 4s 4s/step
Pred shape: (256, 256)
Mask resized shape: (256, 256)
Pred one-hot shape: (256, 256, 7)
Mask one-hot shape: (256, 256, 7)
Metrics for test_01.jpg:
         iou [soft]:
                 bg: 0.8814 [0.8814]
                 brt: 0.0010 [0.0010]
                 ccp/kub: 0.7368 [0.7368]
                 gl: 0.1642 [0.1642]
                 py/mrc: 0.9581 [0.9581]
                 sph: 0.7849 [0.7849]
                 tnt/ttr: 0.0074 [0.0074]
         mean iou [soft]: 0.5048 [0.5048]
         acc: 0.9280

Metrics with void borders for test_01.jpg:
         iou [soft]:
                 bg: 0.8814 [0.8814]
                 brt: 0.0010 [0.0010]
                 ccp/kub: 0.7368 [0.7368]
                 gl: 0.1642 [0.1642]
                 py/mrc: 0.9581 [0.9581]
                 sph: 0.7849 [0.7849]
                 tnt/ttr: 0.0074 [0.0074]
         mean iou [soft]: 0.5048 [0.5048]
         acc: 0.9280

--------------------------------------------------
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step
Pred shape: (256, 256)
Mask resized shape: (256, 256)
Pred one-hot shape: (256, 256, 7)
Mask one-hot shape: (256, 256, 7)
Metrics for test_02.jpg:
         iou [soft]:
                 bg: 0.7972 [0.7972]
                 brt: 1.0000 [1.0000]
                 ccp/kub: 0.1273 [0.1273]
                 gl: 0.1905 [0.1905]
                 py/mrc: 0.9518 [0.9518]
                 sph: 0.5996 [0.5996]
                 tnt/ttr: 0.0000 [0.0000]
         mean iou [soft]: 0.5238 [0.5238]
         acc: 0.8822
```

```
Metrics with void borders for test_02.jpg:
         iou [soft]:
                 bg: 0.7972 [0.7972]
                 brt: 1.0000 [1.0000]
                 ccp/kub: 0.1273 [0.1273]
                 gl: 0.1905 [0.1905]
                 py/mrc: 0.9518 [0.9518]
                 sph: 0.5996 [0.5996]
```