**Методические указания к лабораторной работе № 4**

**Автор: доцент НИТПУ Друки А.А.**

**Семантическая сегментация изображений с использованием архитектуры UNet**

Данная лабораторная работа посвящена семантической сегментации снимков дистанционного зондирования земли на основе модели нейронной сети U-Net.

**Цель работы:** получить навыки сегментации снимков дистанционного зондирования земли на основе модели нейронной сети U-Net.

**Задачи:**

1. Ознакомиться с работой сверточных нейронных сетей в библиотеке Keras;

2. Научиться решать задачу сегментации изображений на основе нейронных сетей.

3. Научиться применять модель сверточной нейронной сети U-Net.

**Модель сверточной нейронной сети U-Net**

**U-Net** – это свёрточная нейронная сеть, которая была создана в 2015 году для сегментации биомедицинских изображений в отделении Computer Science Фрайбургского университета. Архитектура сети представляет собой полносвязную свёрточную сеть, модифицированную так, чтобы она могла работать с меньшим количеством примеров (обучающих образов) и обеспечивала более точную сегментацию.

Сеть содержит сжимающий путь (слева) и расширяющий путь (справа), поэтому архитектура похожа на букву U, что и отражено в названии. На каждом шаге удваивается количество каналов признаков. Сжимающий путь похож на типичную свёрточную сеть, он содержит два подряд свёрточных слоя 3x3, после которых идет слой ReLU и Pooling с функцией максимума 2×2 с шагом 2.

Каждый шаг расширяющего пути содержит слой, обратный операции Pooling, который расширяет карту признаков, после которого следует свертка 2x2, которая уменьшает количество каналов признаков. После идет конкатенация с соответствующим образом обрезанной картой признаков из сжимающего пути и две свертки 3x3, после каждой из которой идет ReLU. На последнем слое свертка 1x1 используется для приведения каждого 64-компонентного вектора признаков до требуемого количества классов. Всего сеть имеет 23 свёрточных слоя.

**Обучающая выборка**

В качестве обучающей и тестовой выборки используется набор изображений Humans in the Loop, который разрабатывался для совместного проекта с Космическим центром Мохаммеда бин Рашида в Дубае, ОАЭ и находится в открытом доступе. Набор данных состоит из аэрофотоснимков Дубая, полученных со спутников MBRSC и аннотированных с помощью семантической сегментации.

Пример входных данных можно увидеть слева на рисунке 1, справа пример семантической сегментации.

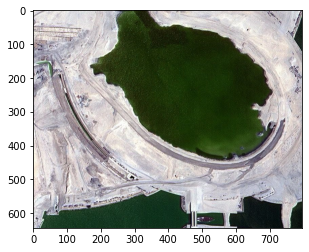
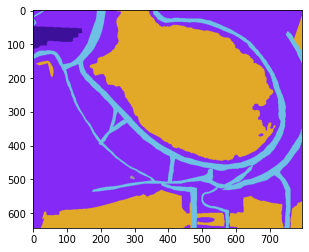
** **

Рисунок 1. Пример изображений для обучения нейронной сети

**Ход работы**

**1. Загрузка данных**

Скачайте выборку изображений по ссылке:

<https://drive.google.com/drive/folders/1eIfEDJjZh3jjOzMVwSk0jbGrlBja_OAt?usp=sharing>

Данная выборка основана на данных с сайта:

<https://humansintheloop.org/uae-dataset-download/?submissionGuid=4d1367f9-97ed-4c5b-b1ff-b6a60749290a>.

**2. Загрузка выборки на Google диск**

Далее Вам требуется загрузить выборку данных на Google диск, для этого зайдите в Google диск (<https://drive.google.com/drive/u/0/my-drive>) и пройдите процедуру авторизации/регистрации (рисунок 2).

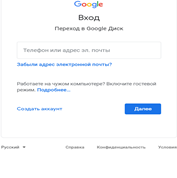


Рисунок 2. Форма регистрации в Google диск

В результате Вы должны увидеть Ваш Google диск (рисунок 3).

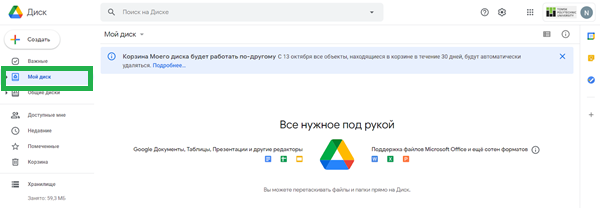


Рисунок 3. Главная страница Google диск

Далее Вам необходимо загрузить выборку изображений на Google диск. Для этого выберите пункт «Создать» -> «Загрузить папку» (рисунок 4).

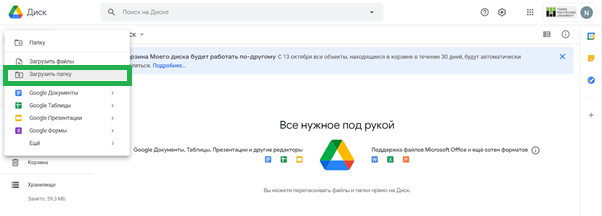


Рисунок 4. Выбор меню «Загрузить папку»

Выберите файл со скачанной выборкой изображений и нажмите кнопку «Загрузить» (рисунок 5).

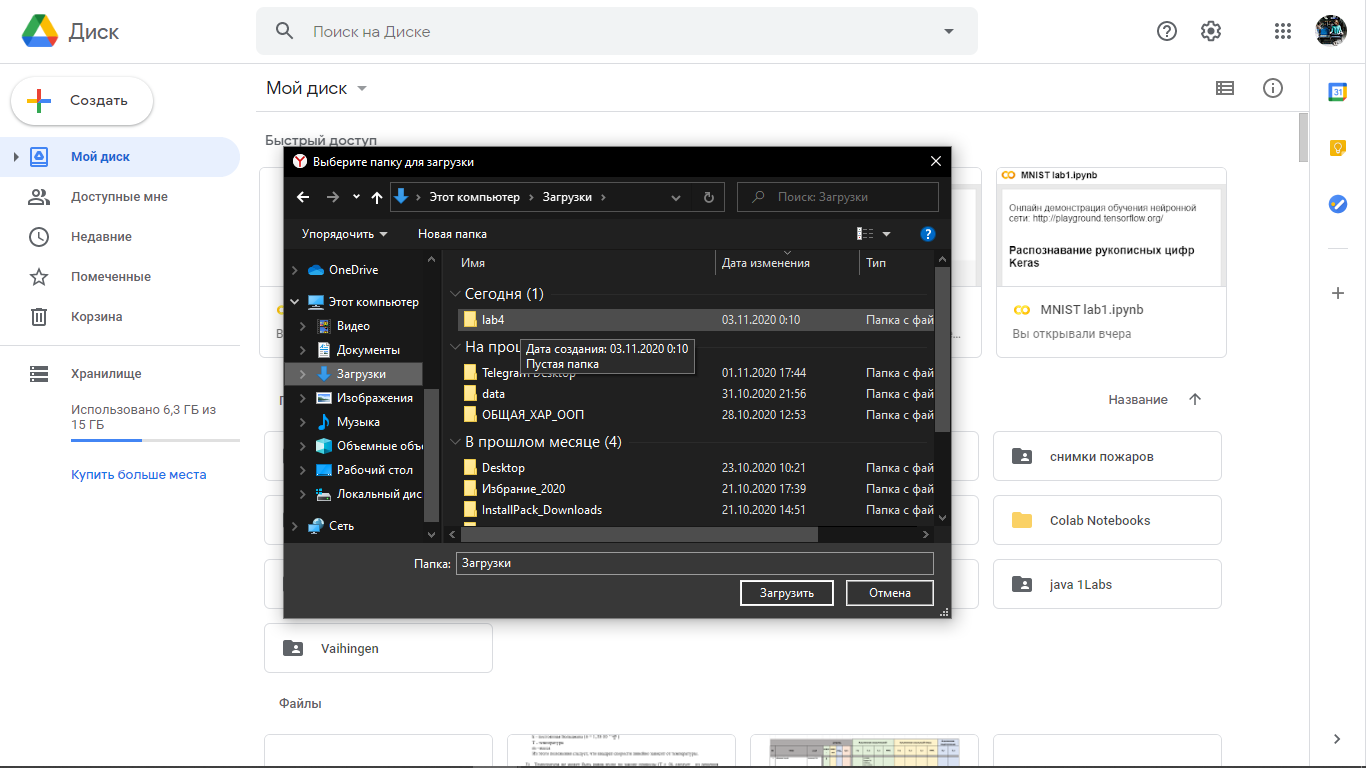


Рисунок 5. Выбор файлов для загрузки

Загруженный файл (lab4) появиться в Вашем Google диске (рисунок 6).

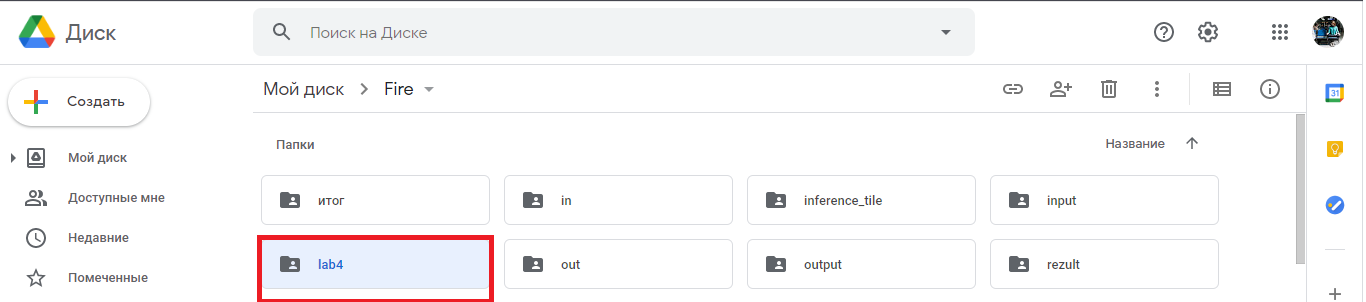


Рисунок 6. Отображение загруженных файлов в Google диске

**3. Подключение к Google диску**

Чтобы подключить Вашу программу к Google диску, вы должны вставить и запустить следующий блок кода:

#подключение к гугл диску

from google.colab import drive

drive.mount('/content/drive')

Запустите код. После запуска появится ссылка доступа, перейдите по ней для авторизации (рисунок 7, 8).

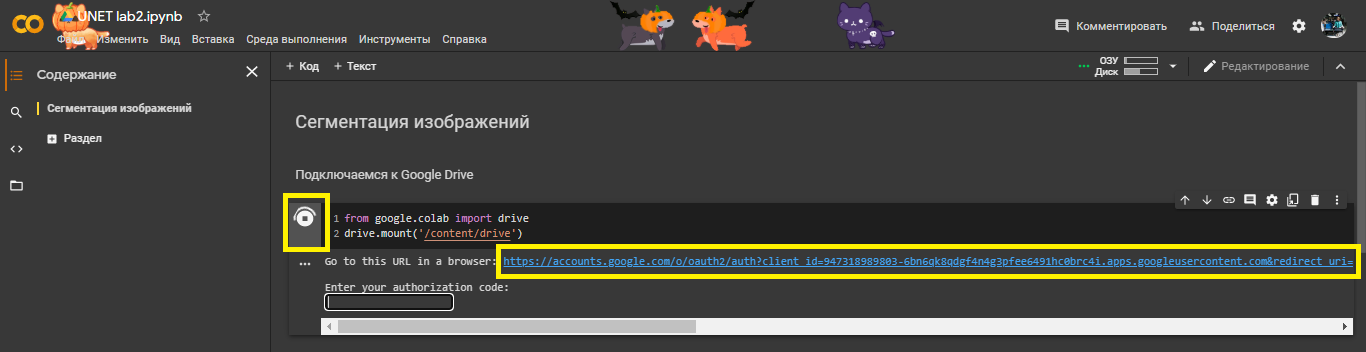


Рисунок 7. Ссылка доступа к Google диску

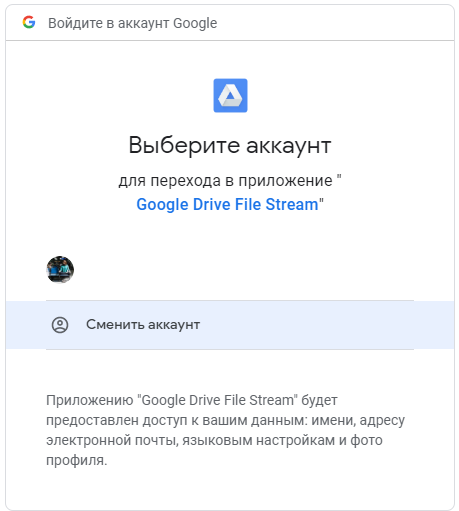


Рисунок 8. Форма авторизации в Google диске

Затем, приложение **«Google Drive File Stream»** запросит разрешение на доступ к вашему аккаунту Google, нажмите «разрешить» (рисунок 9).

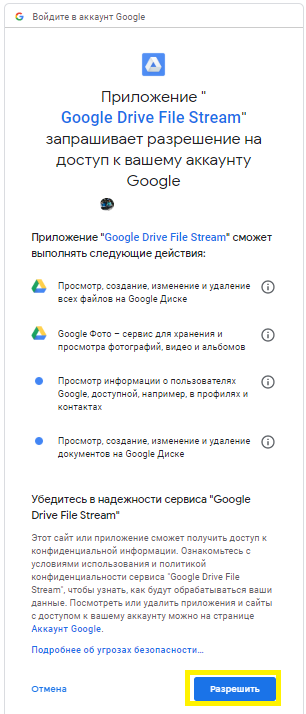


Рисунок 9. Доступ к аккаунту Google

После этого, скопируйте код, перейдите в приложение и вставьте в нужное поле (рисунок 10).

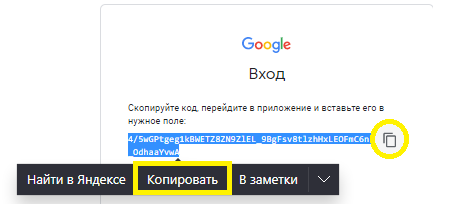


Рисунок 10. Копирование сгенерированного кода

В Google Colab вводим скопированный код и нажимаем Enter (рисунок 11).

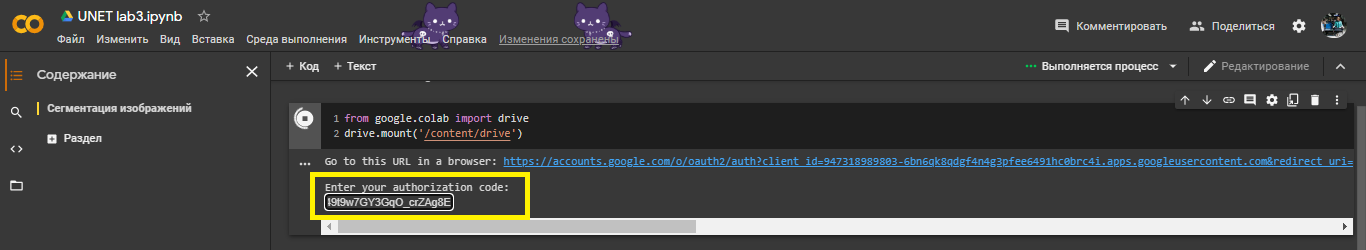


Рисунок 11. Ввод скопированного кода

**4. Импорт данных из Google диска**

Подключим нашу выборку данных используя следующий блок кода:

from PIL import Image

import numpy as np

import matplotlib.pyplot as plt

from skimage import io

import os

def download\_data(path):

data = []

for path\_image in sorted(os.listdir(path=path)):

image = Image.open(path + path\_image) #Открываем изображение.

data.append(np.array(image)) #Загружаем пиксели.

return data

X\_train = download\_data(r"./drive/My Drive/Fire/lab4/train/image/")

Y\_train = download\_data(r"./drive/My Drive/Fire/lab4/train/label/")

X\_test = download\_data(r"./drive/My Drive/Fire/lab4/test/image/")

Y\_test = download\_data(r"./drive/My Drive/Fire/lab4/test/label/")

**NumPy** — это библиотека языка Python, добавляющая поддержку больших многомерных массивов и матриц.

# **Matplotlib** —пользовательская библиотека для построения графиков на языке python.

**scikit-image** — это Python­-пакет с открытым кодом, который работает с массивами NumPy. Он реализует алгоритмы и утилиты для использования в исследовательских, образовательных и промышленных приложениях.

Метод **os.listdir()** в python используется для получения списка всех файлов и каталогов в указанном каталоге.

Данный блок позволяет загрузить изображения выборки данных из Google диска в переменные X\_train, Y\_train, X\_test, Y\_test.

Где:

**X\_train** – Исходные изображения из обучающей выборки данных.

**Y\_train** – Сегментированные исходные изображения из обучающей выборки.

**X\_test** – Исходные изображения из тестовой выборки данных.

**Y\_test** – Сегментированные исходные изображения из тестовой выборки.

**5. Объявление функций предобработки и пост обработки данных**

Вставьте следующий блок кода:

palette = {0 : (60, 16, 152), # Building

1 : (132, 41, 246), # Land

2 : (110, 193, 228), # Road

3 : (254, 221, 58), # Vegetation

4 : (226, 169, 41), # Water

5 : (155, 155, 155)} # Unlabeled

invert\_palette = {v: k for k, v in palette.items()}

# сегментация нейронной сети в RGB изображение

def convert\_to\_color(arr\_2d, palette=palette):

""" Numeric labels to RGB-color encoding """

arr\_3d = np.zeros((arr\_2d.shape[0], arr\_2d.shape[1], 3), dtype=np.uint8)

for c, i in palette.items():

m = arr\_2d == c

arr\_3d[m] = i

return arr\_3d

Атрибут **palette** используется для установки цвета полос. Это помогает различать данные.

Где:

* 0: (60, 16, 152) # Building;
* 1 : (132, 41, 246) # Land;
* 2: (110, 193, 228) # Road;
* 3: (254, 221, 58) # Vegetation;
* 4: (226, 169, 41) # Water;
* 5: (155, 155, 155) # Unlabeled.

Переменная **invert\_palette** позволяет проводить обратное преобразование, цвета в метку класса.

Задаем параметр цвета для определенного класса. Для последующего преобразования сегментированного изображения в приемлемый для нейронной сети вид.

**zeros()** – данная функция возвращает новый массив указанной формы и типа, заполненный нулями.

Метод **items** возвращает множество-подобный (set-like), состоящий из уникальных не повторяющихся элементов, объект типа dict\_items, состоящий из списка кортежей пар ключ-значение.

Вставьте следующий блок кода:

# преобразование сегментированного image in answer network for input in output

def convert\_from\_color(arr\_3d, palette=invert\_palette):

""" RGB-color encoding to grayscale labels """

arr\_2d = np.zeros((arr\_3d.shape[0], arr\_3d.shape[1]), dtype=np.uint8)

arr\_2d = np.zeros((arr\_3d.shape[0], arr\_3d.shape[1]), dtype=np.int8) # принадлежность каждого пикселя классу

min\_distance = np.zeros((arr\_3d.shape[0], arr\_3d.shape[1]), dtype=np.float32)+1000 # расстояние до ближайшего класса для пикселей

for c, i in palette.items():

distance = np.sum((arr\_3d - np.array(c).reshape(1, 1, 3))\*\*2, axis=-1)\*\*(1/2) # ищем расстояние для каждого пикселя до проверяемого класса по евклиду рас-ие

condition = min\_distance > distance # поиск элементов меньше min\_distance

min\_distance[condition] = distance[condition] # замена дистанции найденных элементов

arr\_2d[condition] = i # замена класса найденных элементов

for c, i in palette.items():

m = np.all(arr\_3d == np.array(c).reshape(1, 1, 3), axis=2)

arr\_2d[m] = i

arr\_2d = arr\_2d.tolist()

for i in range(len(arr\_2d)):

for j in range(len(arr\_2d[0])):

label = [0, 0, 0, 0, 0, 0]

label[arr\_2d[i][j]] = 1

arr\_2d[i][j] = label

arr\_2d = np.array(arr\_2d)

return arr\_2d

Данная функция преобразует результат сегментации нейронной сети в RGB изображение, удобное для восприятия пользователя.

**6. Предобработка исходных изображений и сегментированных изображений в ответ сети**

Следующий блок кода подготавливает данные для использования нейронной сетью:

“””X\_train\_pred = X\_train/255

X\_test\_pred = X\_test/255”””

X\_train\_pred = np.array(X\_train).reshape([7, 644, 796, 3])/255

X\_test\_pred = np.array(X\_test).reshape([2, 644, 796, 3])/255

Y\_train\_pred = []

for i in range(len(Y\_train)):

Y\_train\_pred.append(convert\_from\_color(Y\_train[i][:, :, :3]))

Y\_train\_pred = np.array(Y\_train\_pred)

Y\_test\_pred = []

for i in range(len(Y\_test)):

Y\_test\_pred.append(convert\_from\_color(Y\_test[i][:, :, :3]))

Y\_test\_pred = np.array(Y\_test\_pred)

В результате выполнения данного кода в переменные X\_train\_pred, Y\_train\_pred, X\_test\_pred, Y\_test\_pred будут записаны данные после предварительной обработки, где:

**X\_train\_pred** – Исходные изображения из обучающей выборки данных.

**Y\_train\_pred** – Сегментированные исходные изображения из обучающей выборки.

**X\_test\_pred** – Исходные изображения из тестовой выборки данных.

**Y\_test\_pred** – Сегментированные исходные изображения из тестовой выборки.

**range()** –позволяет Вам генерировать ряд целых чисел в рамках заданного диапазона.

**len()** – данная функция возвращает целое число (**int**) равное количеству элементов объекта/массива.

**7. Объявление топологии нейронной сети**

Чтобы объявить нейронную сеть U-Net (используется упрощенный вариант данной нейронной сети с меньшим количеством слоев), вставьте следующую часть кода:

import numpy as np

from keras.models import \*

from keras.layers import \*

from keras.optimizers import \*

… Добавьте свой код …

**NumPy** – это библиотека языка Python, добавляющая поддержку больших многомерных массивов и матриц.

**ReLU** – функция активации, выполняющая простое нелинейное преобразование поданных на вход данных (x). Возвращает x, если x > 0 и 0 в противном случае. Отличается высокой скоростью вычисления.

**Softmax** – функция активации, выполняющая нелинейное преобразование на основе математической функции: взвешенная сигмоида. Также может быть использована как функция активации для выходного слоя в задачах многоклассовой классификации. Преобразует значения в вектор вероятностей.

**Conv2D** –2D Convolution Layer, этот слой создает ядро свертки, которое заполняется входными слоями, что помогает получить тензор на выходе. При создании слоя необходимо передать следующие параметры Conv2D(units, kernel\_size, input\_shape=None, activation=None, padding = 'same'), где **units** – количество нейронов, **kernel\_size** – размер ядра свертки, **input\_shape** – задание формы входных данных (например, размер изображения и количество каналов изображения), **activation** – функция активации, **padding** – параметр "same" означает не изменять размер изображения после его обработки сверточным слоем.

**Optimizer** – (функция оптимизатора) – оптимизационный алгоритм, задача которого – изменение весов сети для уменьшения ошибки сети в процессе обучения.

**Adam** – оптимизационный алгоритм, используемый для обучения нейронной сети, основная функция оптимизатора – изменение весов для уменьшения ошибки сети в процессе обучения. Для каждого нейрона алгоритм изменяет веса индивидуально.

**8. Обучение нейронной сети**

Напишем следующую строчку кода, которая будет отвечать за обучение нейронной сети:

… Добавьте свой код …

**9. Тестирование нейронной сети**

Вставляем данный код в отдельный блок, он отвечает за тестирование нашей нейронной сети:

… Добавьте свой код …

**10. Запускаем нейронную сеть для сегментации изображений тестовой выборки**

Чтобы сегментировать изображения, запустите следующий код:

out = model.predict(X\_test\_pred, batch\_size=1)

**11. Вывод предсказания нейронной сети для выбранного изображения**

Чтобы вывести результат сегментации запустите следующий блок кода:

I = 0 # номер картинки после обработки нейронной сетью

plt.imshow(np.array(X\_test)[I])

plt.show()

plt.imshow(np.array(Y\_test)[I])

plt.show()

plt.imshow(convert\_to\_color(np.argmax(out[I], axis=-1)))

plt.show()

Задав номер изображения **(I)** для прогнозирования мы можем получить ответ нейронной сети на экран.

**imshow()** – функция для вывода изображения на экран.

**cmap** – выбор цветовой палитры выводимого изображения (например, в оттенках серого).

**reshape()** – изменение формы массива.

**predict()** – запускает работу нейронной сети для предсказания / распознавание изображения.

**argmax()** – вывод индекса максимального элемента массива.

На рисунках 12-14 представлены результаты сегментации исходного изображения.

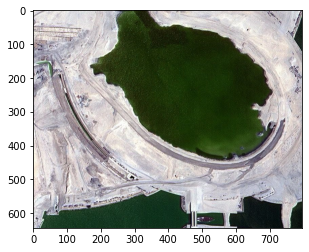
 ****

Рисунок 12. Пример исходных изображений

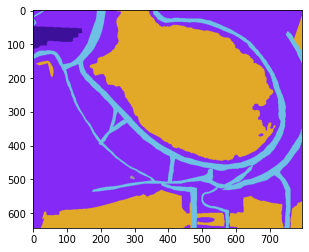
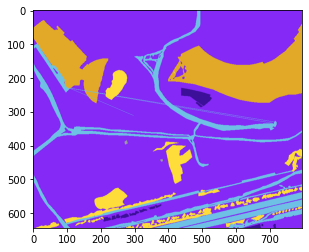
 

Рисунок 13. Желаемый результат сегментации

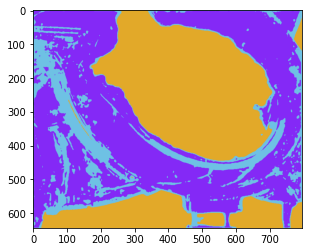
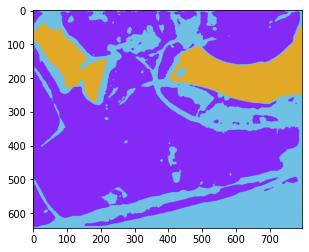
 

Рисунок 14. Полученный результат сегментации

**Вывод**

В результате выполнения данной лабораторной работы Вы получили навыки сегментации снимков дистанционного зондирования земли на основе модели нейронной сети U-Net.

**Критерии оценки**

Максимальное число баллов, которое можно получить по лабораторной работе – 8 баллов.

Минимальное число баллов, при котором задание засчитывается – 5 баллов.

**Выполнение задания**

Выполнено меньше половины задания – 0 баллов.

Большей частью выполнено – 2 балл.

Выполнено полностью – 4 балла.

**Качество отчета**

Сильные огрехи в оформлении или удовлетворяет менее половины требований к отчету – 0 баллов.

Хорошее оформление. Удовлетворяет большей части требований – 2 балл.

Соответствует всем требованиям – 4 балла.