МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Инженерная школа информационных технологий и робототехники

Отделение информационных технологий

09.04.01 «Информатика и вычислительная техника»

Отчет по практической работе №3
«Сегментация изображений»

МАШИННОЕ ОБУЧЕНИЕ

Исполнитель:

студент группы 8ВМ22 Ямкин Н.Н. 10.09.2023

Руководитель:

Доцент (ОИТ, ИШИТР) Друки А.А.

Томск – 2023

Содержание

1. Цель работы

Получить навыки сегментации изображений с помощью сверточных нейронных сетей в библиотеке машинного обучения Keras.

2. Задачи

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

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

3. Ход работы

Подключаем необходимые библиотеки:

```
import numpy as np
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.layers import *
import matplotlib.pyplot as plt
from PIL import Image
import os.path
```

Подключаем Google-диск:

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

Определяем функцию загрузки изображений:

```
def from_png_to_numpy(folder_path):
  list_of_filenames = os.listdir(folder_path)
  data = []
  for i in range(len(list_of_filenames)):
    filename = folder_path+'/'+str(i)+'.png'
    img = Image.open(filename)
    img_to_np = np.array( img, dtype='uint8' )
    data.append(img_to_np)
  return np.array(data)
```

Загружаем исходный набор данных:

```
x_train = from_png_to_numpy('/content/drive/MyDrive/Клеточные
стенки/membrane/train/image')
```

```
y_train = from_png_to_numpy('/content/drive/MyDrive/Клеточные
стенки/membrane/train/mask')
x_test = from_png_to_numpy('/content/drive/MyDrive/Клеточные
стенки/membrane/test/image')
```

Отображаем изображение из обучающей выборки:

```
# Проверяем отображение
plt.imshow(x_train[3], cmap = 'gray')
```

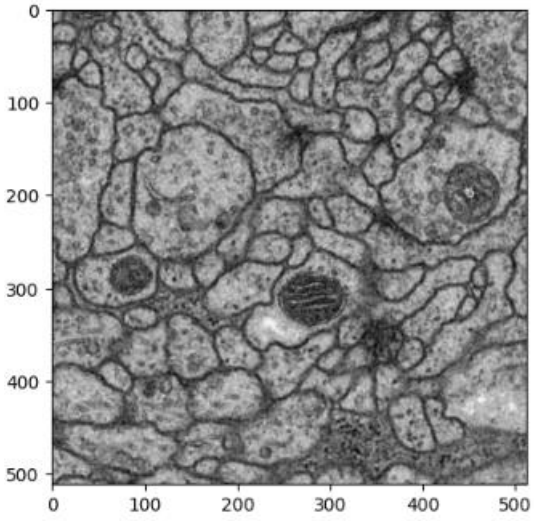

Рисунок 1 – Исходное изображение

Отображаем соответствующее сегментированное изображение из обучающей выборки:

```
plt.imshow(y_train[3],cmap = 'gray')
```

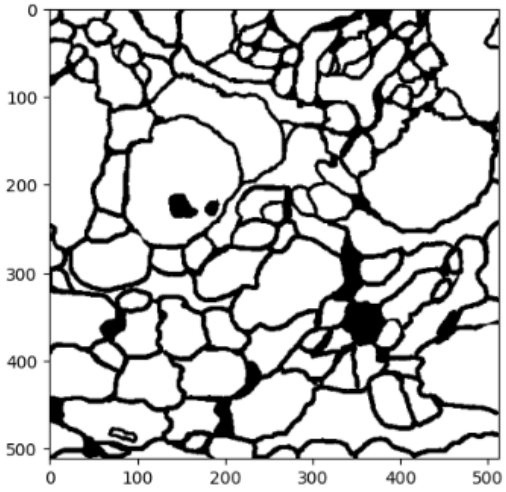

Рисунок 2 – Сегментированное изображение

Нормализуем исходный набор данных:

```
# нормализация данных (приводим значения пикселей в интервал [0, 1])
x_train = x_train / 255
y_train = y_train / 255
x_test = x_test / 255
```

Создадим нейронную сеть:

```
input_shape = (512,512,1)

model = keras.Sequential(
    [
        keras.Input(shape=input_shape),
        # Conv2D - этот слой создает ядро свертки
        # units - количество ядер свертки на каждом слое (16),
        # kernel_size - размер ядра свертки,
        # input_shape - задание формы входных данных,
        # activation - функция активации,
        #  padding - параметр "same" означает не изменять размер
изображения после его обработки сверточным слоем (добавляются нули вокруг
изображения для сохранения размера)
        layers.Conv2D(16, kernel_size=(5,5), activation="relu",
padding="same"),
        layers.Conv2D(16, kernel_size=(5,5), activation="relu",
padding="same"),
        layers.Conv2D(16, kernel_size=(5,5), activation="relu",
padding="same"),
        layers.Conv2D(16, kernel_size=(5,5), activation="relu",
padding="same"),
        layers.Conv2D(16, kernel_size=(5,5), activation="relu",
padding="same"),

        layers.Conv2D(16, kernel_size=(5,5), activation="relu",
padding="same"),
        layers.Conv2D(16, kernel_size=(5,5), activation="relu",
padding="same"),
        layers.Conv2D(16, kernel_size=(5,5), activation="relu",
padding="same"),
        layers.Conv2D(16, kernel_size=(5,5), activation="relu",
padding="same"),
        layers.Conv2D(16, kernel_size=(5,5), activation="relu",
padding="same"),

        layers.Conv2D(16, kernel_size=(5,5), activation="relu",
padding="same"),
        layers.Conv2D(16, kernel_size=(5,5), activation="relu",
padding="same"),
        layers.Conv2D(16, kernel_size=(5,5), activation="relu",
padding="same"),
```

```
        layers.Conv2D(16, kernel_size=(5,5), activation="relu",
padding="same"),
        layers.Conv2D(16, kernel_size=(5,5), activation="relu",
padding="same"),

        layers.Conv2D(1, kernel_size=(5,5), activation="sigmoid",
padding="same"),
    ]
)

model.summary()
```

Основная часть сети состоит из последовательности сверточных слоев, с применением функции активации «relu» и с одинаковым размером фильтра (5x5) и количеством фильтров (16) для каждого слоя. Всего в сети 14 сверточных слоев.

После последнего сверточного слоя применяется слой Conv2D с 1 фильтром и размером фильтра (5x5) с сигмоидальной функцией активации. Этот слой возвращает результат в виде одного значения, которое интерпретируется как вероятность принадлежности каждого пикселя к классу 1.

Каждый фильтр представляет собой матрицу весов, которая перемещается по всему входному изображению для выполнения операции свертки. Учитывая размер фильтра (например, 5x5), он скользит по изображению, выполняет покоординатное умножение значений пикселей на соответствующие элементы фильтра и суммирует результат.

Эта операция свертки выполняется на всех пикселях изображения, за исключением краев (если используется режим заполнения «same»). В результате каждого прохода операции свертки образуется одно числовое значение, называемое активацией или картой признаков.

Каждый фильтр будет обрабатывать входное изображение, но веса фильтров инициализируются случайным образом перед обучением и затем оптимизируются в процессе обучения.

Активация на каждом пикселе карты признаков сохраняется и передается следующему слою для дальнейшей обработки. Это позволяет нейронной сети сосредоточиться на выделении различных характеристик изображения, таких как границы, текстуры, формы и т. д., на каждом последующем слое.

Обучим нейронную сеть:

```
batch_size = 1
epochs = 15
model.compile(loss="MSE", optimizer="adam", metrics=["accuracy"])
model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs,
validation_split=0.1)
```

Проверим работу нейронной сети на тестовой выборке:

```
y_pred = model.predict(x_test)
```

Выведем предсказание нейронной сети:

```
plt.imshow(y_pred[1], cmap='gray')
plt.show()
```

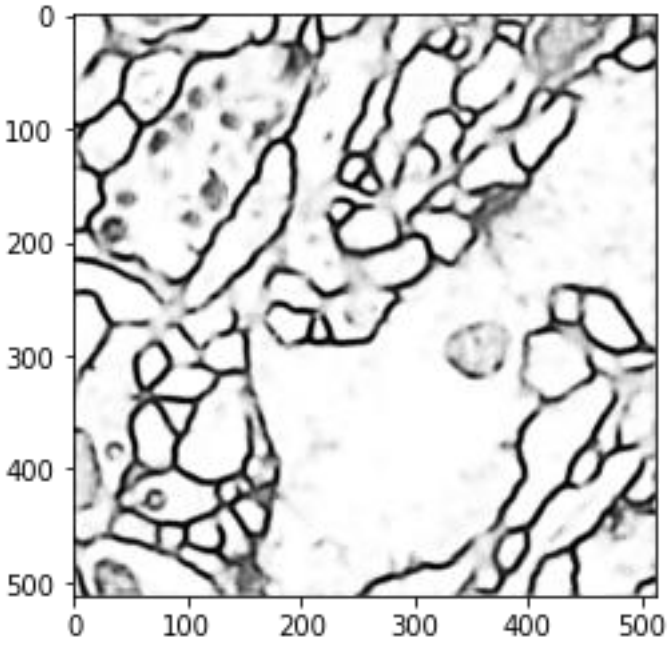

Рисунок 3 – Предсказание нейронной сети

Выведем 8 сегментированных нейросетью изображений:

```
plt.figure(figsize=(6,6))
```

```
for i in range(8):
    plt.subplot(4,4,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(y_pred[i], cmap='gray')
```

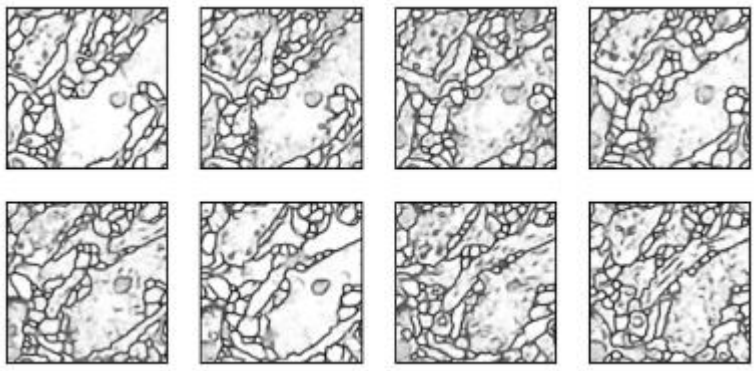

Рисунок 4 – Предсказание нейронной сети

4. Заключение

В результате выполнения данной лабораторной работы были получены навыки сегментации изображений с помощью свёрточных нейронных сетей с использованием библиотеки машинного обучения Keras. Также были получены навыки реализации топологии сверточной нейронной сети, её обучения и тестирования.