МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Центр цифровых
образовательных технологий

09.04.01 Информатика и вычислительная техника

Сегментация изображений с помощью свёрточных нейронных сетей (CNN)
Вариант 2

ЛАБОРАТОРНАЯ РАБОТА № 6

по дисциплине:
Машинное обучение

Исполнитель:

студент группы 8ВМ42 Текере Ричард

Руководитель:

доцент ОИТ, ИШИТР Друки А.А.

Томск - 2025

ВВЕДЕНИЕ

В компьютерном зрении, сегментация – это процесс разделения цифрового изображения на несколько сегментов (множество пикселей). Другими словами, это процесс присвоения таких меток каждому пикселю изображения, что пиксели с одинаковыми метками имеют общие визуальные характеристики. Сегментация изображений обычно используется для того, чтобы выделить объекты на изображениях.Результатом сегментации изображения является множество сегментов, которые вместе покрывают всё изображение. В пределах одного сегмента пикселям придаются схожие визуальные характеристики, например, цвет, яркость или текстура.Сегодня обычно сегментация выполняется с помощью сверточных нейронных сетей.

Сверточная нейронная сеть (convolutional neural network, CNN) – специальная архитектура искусственных НС, которая имеет двумерную структуру и нацеленная на работу с изображениями. Предложена французским ученым Яном Лекуном в 1998 году.

Цель работы

Получить навыки сегментации изображений с помощью свёрточных нейронных сетей в среде программирования Google Colab с использованием Keras.Задачи работы:

1. Изучить работу свёрточных нейронных сетей (CNN) в Keras.
2. Научиться загружать и предобрабатывать данные.
3. Реализовать простую свёрточную нейронную сеть.
4. Провести обучение и тестирование модели.
5. Получить результаты сегментации изображений.

Ход работы

1. Загрузка данных

В качестве данных для обучения и тестирования используются изображения клеточных мембран (аналогично ISBI-2012). Набор данных разбит на обучающую и тестовую выборки.

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

from PIL import Image
import numpy as np
import os

def download_data(path):
    data = []
    for path_image in sorted(os.listdir(path=path)):
        image = Image.open(path + path_image)
        data.append(np.array(image)[:640, :352])
    return data

X_train = download_data(r"./drive/My Drive/membrane/train/image/")
Y_train = download_data(r"./drive/My Drive/membrane/train/mask/")
X_test = download_data(r"./drive/My Drive/membrane/test/image/")
Y_test = download_data(r"./drive/My Drive/membrane/test/mask/")
```

Вывод:

```
1 I = 10 # номер изображения для вывода
2 plt.imshow(X_train[I], cmap='gray') # вывод изображения на экран
```

```
<matplotlib.image.AxesImage at 0x7feeb3a10d30>
```

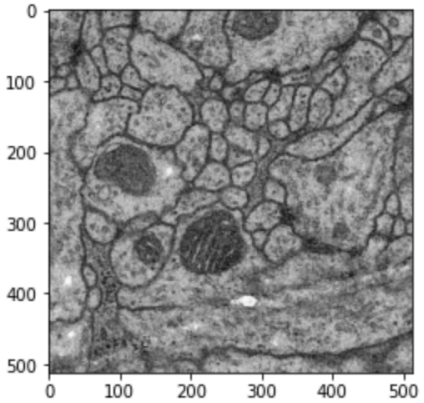

Рисунок 1 — Примеры исходных изображений

2. Предобработка данных

Перед обучением необходимо провести нормализацию данных и изменить их форму:

```
X_train_pred = np.array(X_train).reshape([30, 512, 512, 1]) / 255
Y_train_pred = np.array(Y_train).reshape([30, 512, 512, 1]) / 255

X_test_pred = np.array(X_test).reshape([30, 512, 512, 1]) / 255
Y_test_pred = np.array(Y_test).reshape([30, 512, 512, 1]) / 255
```

3. Создание модели нейронной сети

Реализуем простую последовательную модель с несколькими свёрточными слоями:

```
from keras.models import Sequential
from keras.layers import Conv2D
from keras.optimizers import Adam

model = Sequential()
model.add(Conv2D(64, 5, input_shape=[512, 512, 1], activation='relu',
padding='same'))
model.add(Conv2D(64, 5, activation='relu', padding='same'))
model.add(Conv2D(64, 5, activation='relu', padding='same'))
model.add(Conv2D(64, 5, activation='relu', padding='same'))
model.add(Conv2D(64, 5, activation='relu', padding='same'))
model.add(Conv2D(1, 1, activation='sigmoid'))

model.compile(optimizer=Adam(), loss='mse')
```

Вывод:

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 512, 512, 64)	1,664
conv2d_1 (Conv2D)	(None, 512, 512, 64)	102,464
conv2d_2 (Conv2D)	(None, 512, 512, 64)	102,464
conv2d_3 (Conv2D)	(None, 512, 512, 64)	102,464
conv2d_4 (Conv2D)	(None, 512, 512, 64)	102,464
conv2d_5 (Conv2D)	(None, 512, 512, 1)	65

Total params: 411,585 (1.57 MB)
Trainable params: 411,585 (1.57 MB)
Non-trainable params: 0 (0.00 B)

Рисунок 2 — Сводная информация о слоях модели

4. Обучение нейронной сети

Проведём обучение модели на подготовленных данных:

```
model.fit(X_train_pred, Y_train_pred, epochs=15, batch_size=1)
```

Вывод:

```
Epoch 1/15
 2/30 [=>............................] - ETA: 2s - loss: 0.2099WARNING:tensorflow:Callbacks
30/30 [==============================] - 5s 179ms/step - loss: 0.1572
Epoch 2/15
30/30 [==============================] - 6s 185ms/step - loss: 0.1255
Epoch 3/15
30/30 [==============================] - 6s 185ms/step - loss: 0.1018
Epoch 4/15
30/30 [==============================] - 6s 185ms/step - loss: 0.1052
Epoch 5/15
30/30 [==============================] - 6s 185ms/step - loss: 0.0953
Epoch 6/15
30/30 [==============================] - 6s 185ms/step - loss: 0.0924
Epoch 7/15
30/30 [==============================] - 6s 185ms/step - loss: 0.0851
Epoch 8/15
30/30 [==============================] - 6s 185ms/step - loss: 0.0849
Epoch 9/15
30/30 [==============================] - 6s 185ms/step - loss: 0.0851
Epoch 10/15
30/30 [==============================] - 6s 185ms/step - loss: 0.0812
Epoch 11/15
30/30 [==============================] - 6s 185ms/step - loss: 0.0800
Epoch 12/15
30/30 [==============================] - 6s 185ms/step - loss: 0.0787
Epoch 13/15
30/30 [==============================] - 6s 185ms/step - loss: 0.0792
Epoch 14/15
30/30 [==============================] - 6s 185ms/step - loss: 0.0795
Epoch 15/15
30/30 [==============================] - 6s 185ms/step - loss: 0.0801
<tensorflow.python.keras.callbacks.History at 0x7fee704d86a0>
```

Рисунок 3 — Процесса обучения.

6. Сегментация тестовых изображений

Выполним сегментацию изображений тестовой выборки:

```
out = model.predict(X_test_pred, batch_size=1)

I = 0
plt.imshow(out[I].reshape([512, 512]), cmap='gray')
```

Вывод:

```
<matplotlib.image.AxesImage at 0x7fee702976a0>
```

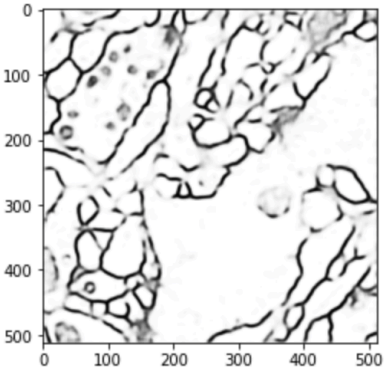

Рисунок 4 — Результата сегментации

Заключение

В ходе лабораторной работы были приобретены практические навыки разработки и обучения свёрточных нейронных сетей для задачи сегментации изображений. Рассмотренная модель продемонстрировала способность эффективно выделять структуры на изображениях, восстанавливая важную информацию и улучшая качество визуального восприятия данных.

Приложение А

Google _Colab_Link

```
from google.colab import drive
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
import os
from keras.models import Sequential
from keras.layers import Conv2D
from keras.optimizers import Adam

# Подключение к Google Drive
drive.mount('/content/drive')

# Загрузка данных
def download_data(path):
    data = []
    for path_image in sorted(os.listdir(path=path)):
        image = Image.open(path + path_image)
        data.append(np.array(image)[:640, :352])
    return data

X_train = download_data(r"./drive/My Drive/membrane/train/image/")
Y_train = download_data(r"./drive/My Drive/membrane/train/mask/")
X_test = download_data(r"./drive/My Drive/membrane/test/image/")
Y_test = download_data(r"./drive/My Drive/membrane/test/mask/")

# Предобработка данных
X_train_pred = np.array(X_train).reshape([30, 512, 512, 1]) / 255
Y_train_pred = np.array(Y_train).reshape([30, 512, 512, 1]) / 255
X_test_pred = np.array(X_test).reshape([30, 512, 512, 1]) / 255
Y_test_pred = np.array(Y_test).reshape([30, 512, 512, 1]) / 255

# Создание модели
model = Sequential()
model.add(Conv2D(64, 5, input_shape=[512, 512, 1], activation='relu', padding='same'))
model.add(Conv2D(64, 5, activation='relu', padding='same'))
model.add(Conv2D(64, 5, activation='relu', padding='same'))
model.add(Conv2D(64, 5, activation='relu', padding='same'))
model.add(Conv2D(64, 5, activation='relu', padding='same'))
model.add(Conv2D(1, 1, activation='sigmoid'))
model.compile(optimizer=Adam(), loss='mse')

# Обучение модели
model.fit(X_train_pred, Y_train_pred, epochs=15, batch_size=1)

# Тестирование модели
print("MSE:", model.evaluate(X_test_pred, Y_test_pred, batch_size=1))
```

```
# Сегментация и вывод результата
out = model.predict(X_test_pred, batch_size=1)
I = 0
plt.imshow(out[I].reshape([512, 512]), cmap='gray')
plt.show()
```