


## Оглавление

1 Описание задачи.....	2
2 Opensv.....	3
2.1 Описание работы программы .....	3
2.2 Анализ точности.....	19
3 Нейронная сеть .....	21
3.1 Описание работы программы .....	21
3.2 Анализ точности.....	31
4 MatchTemplate.....	33
5 Выводы .....	36

# 1 Описание задачи

Дана фотография книжной полки. Требуется посчитать количество корешков книг из разных коллекций.

Всего 7 коллекций. Ниже представлены примеры соответствующих книг:

Librarium	Великая поэзия	Азбука Классика	Всемирная литература	Попадан-чество	Любовь. Интрига. Тайна	Николай Стариков
						















## 2 Openscv

### 2.1 Описание работы программы

Для работы были сделаны 22 фотографии книжных полок (bs1.JPG, ..., bs22.JPG) и таблица Excel (books data.xlsx), содержащая следующие столбцы:

Столбец А	Столбец В	Столбец С
Фрагмент изображения корешка книги, преобразованный к размеру 32×32	Идентификатор коллекции (цифра от 0 до 6 вкл.)	Название коллекции

Для каждой коллекции сделано 13 таких строк. Ниже приведен фрагмент таблицы:

	А	В	С
1			4 Всемирная Литература
2			1 Великая Поэзия
3			3 Попаданчество
4			5 Любовь. Интрига. Тайна
5			0 Librarium
6			4 Всемирная Литература
7			5 Любовь. Интрига. Тайна
8			2 Азбука Классика
9			5 Любовь. Интрига. Тайна
10			5 Любовь. Интрига. Тайна
11			3 Попаданчество
12			0 Librarium
13			5 Любовь. Интрига. Тайна
14			1 Великая Поэзия

В данном случае эти файлы лежат в папке (bookshelves) на Google-диске. Поэтому в начале программы следует к нему подключиться:

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

Mounted at /content/drive
```

Для работы с Excel устанавливаем openpyxl и openpyxl-image-loader:

```
!pip3 install openpyxl
!pip3 install openpyxl-image-loader
```

Импортируем необходимые модули:

```
# для работы с массивами
import numpy as np
# opencv для работы с изображениями
import cv2
from google.colab.patches import cv2_imshow
# для работы с Excel
import openpyxl
from openpyxl_image_loader import SheetImageLoader
```

Загружаем изображение для анализа с помощью метода `cv2.imread()`, в скобках указывая путь к нужному файлу. Выводим на экран размеры и само фото книжной полки. Например:

```
image = cv2.imread("/content/drive/MyDrive/bookshelves/bs20.JPG")

height, width = image.shape[:2]
# shape это (rows, columns, other) - берем только первые 2 для высоты и ширины

print("%sx%s" % (width, height)) # печатаем размеры

cv2_imshow(image) # показать изображение в окне
```



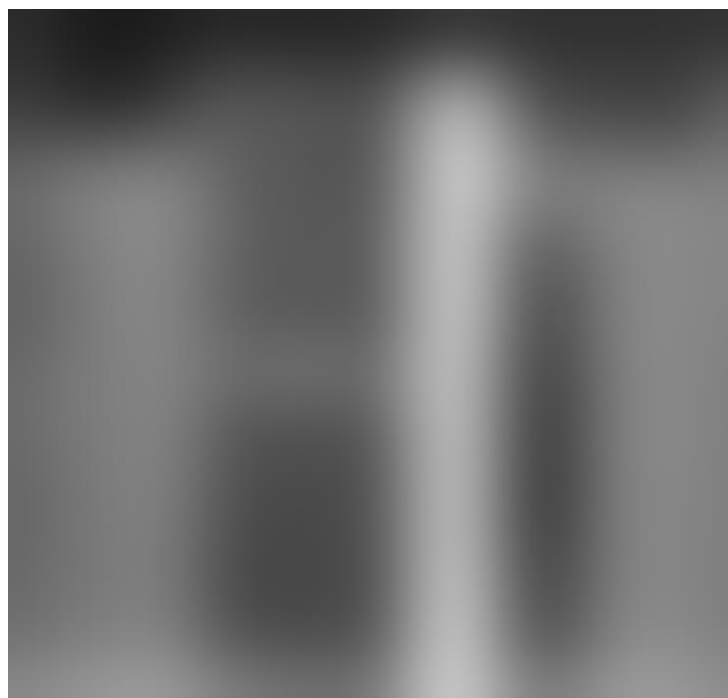
Преобразуем изображение в градации серого:

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```



Сильно размываем фильтром Гаусса:

```
blur = cv2.GaussianBlur(gray, (0,0), sigmaX=40, sigmaY=40)
```



Делим серое изображение на сильно размытое – это «уберет фон», сделает изображение светлее, но оставит наиболее четкие границы:

```
divide = cv2.divide(gray, blur, scale=255)
```



Размываем результат фильтром Гаусса. Выбрано прямоугольное ядро, для того, чтобы сохранить больше вертикальных линий:

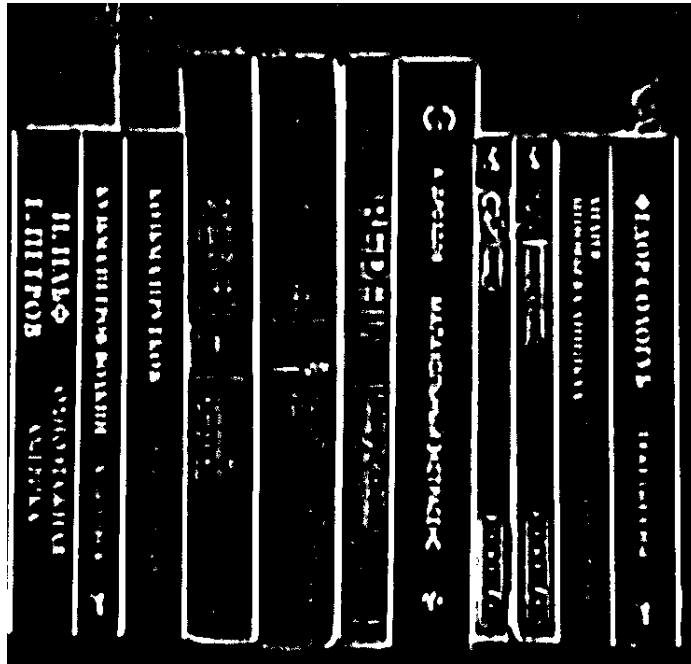
```
blurred = cv2.GaussianBlur(divide, (17, 21), 0)
```



Применяем пороговое преобразование и получаем изображение с выделенными границами:



```
thresh_mean = cv2.adaptiveThreshold(blurred,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY_INV,11,3)
```



На этом завершается подготовка исходного изображения к анализу. Видно, что линии, разделяющие книги, легко различимы, а надписи на книгах частично исчезли.

Создаём объект `workbook` для работы с Excel-таблицей и выбираем нужный лист:

```
x1 = openpyxl.load_workbook('/content/drive/MyDrive/bookshelves/books_data.xlsx')  
sheet = x1['Лист1']
```

Создаём объект для получения изображений из Excel:

```
image_loader = SheetImageLoader(sheet)
```

Создаём массив с названиями коллекций книг (индексы соответствуют идентификаторам коллекций в столбце В таблицы) и пустой массив для хранения средних цветов образцов в формате RGB:

```
# названия коллекций книг  
names = ["Librarium", "Великая поэзия", "Азбука Классика", "Попаданчество",  
         "Всемирная литература", "Любовь. Интрига. Тайна", "Николай Стариков"]  
  
# массив для цветов образцов  
colours = np.zeros((len(names),3)) # RGB
```

В цикле по строкам таблицы `bookset opencv.xlsx` в каждый элемент `colours` записывается сумма средних значений цветов образцов книг из соответствующей коллекции:

- с помощью метода `get` объекта `image_loader` получаем изображение из ячейки столбца A
- преобразуем в массив `numpy`
- используя метод `cv2.mean`, считаем средний цвет
- прибавляем его к элементу `colours`, индекс которого равен идентификатору из столбца B (получаем с помощью метода `value`)

После выхода из цикла делим элементы массива на количество книг каждой коллекции для получения средних цветов коллекций:

```
n = 13 # кол-во образцов каждой книги

# в каждый элемент colours записывается сумма средних значений цветов
# образцов книг из соответствующей коллекции
for i in range(1, 92):
    colours[int((sheet['B'+str(i)]).value)] += (cv2.mean(np.array(image_loader.get('A'+str(i))))[:3])
colours = colours/n
```

Выводим на экран названия коллекций и соответствующие средние цвета:

```
for i in range (0,7):
    print(names[i])
    print(colours[i])
    print('')
```

```
Librarium
[175.14873798 132.66932091 106.42127404]

Великая поэзия
[128.65429688 78.94005409 63.1190655 ]

Азбука Классика
[60.51427284 78.59269832 64.32271635]

Попаданчество
[230.14317909 232.86666166 236.79852764]

Всемирная литература
[ 89.03891226 91.34788161 101.7105619 ]

Любовь. Интрига. Тайна
[72.04867788 64.02065805 58.27328726]

Николай Стариков
[239.0878155 59.06527945 67.74737079]
```

Создаём массив, в котором будем хранить отклонения значения цвета рассматриваемой книги от среднего цвета каждой коллекции:



```
stat = np.zeros(7)
```

Указываем параметры, которые будут переданы в метод Хафа для обнаружения линий на изображении. Параметры выбраны эмпирически:

```
# изображение для рисования линий
lines_output = image.copy()

#### параметры для линий Хафа ####

# пороговое изображение
edges = thresh_mean
# разрешение сетки Хафа в пикселях
rho = 1
# Разрешение сетки Хафа в радианах
theta = np.pi / 135
# Минимальное количество точек пересечения для обнаружения линии
threshold = 35
# Минимальная длина линии в пикселях
min_line_length = height / 4
# Максимальное расстояние в пикселях между соединяемыми сегментами линии
max_line_gap = 2 * max(width, height) / min(width, height)
# Холст, на котором будет рисовать линии
line_image = np.copy(lines_output) * 0
```

Применяем преобразование Хафа к пороговому изображению, полученному ранее, с помощью метода `cv2.HoughLinesP`. В отличие от метода `cv2.HoughLines`, возвращающего полярные координаты, `cv2.HoughLinesP` возвращает декартовы координаты концов обнаруженных линий:

```
# Применяем преобразование Хафа
# Результат - массив координат концов обнаруженных линий
lines = cv2.HoughLinesP(
    edges,
    rho,
    theta,
    threshold,
    np.array([]),
    min_line_length,
    max_line_gap
)
```

Вводим счетчики для границ и для книг и переменные для координат предыдущей линии:

```
count = 0    # счетчик книг
k = 0        # счётчик линий Хафа

# координаты предыдущей линии
prevx1 = 0
prevy2 = 0
prevx2 = 0
prevy1 = 0
```

Сортируем линии в порядке возрастания координаты x (слева направо):

```
lines = sorted(lines, key=lambda line: line[...][0])
```

Организуем цикл по обнаруженным линиям, пропуская линии, близкие к горизонтальным:

```
# цикл по линиям Хафа
if lines is not None:
    for line in lines:
        # координаты концов линии
        for x1,y1,x2,y2 in line:
            if abs(y1-y2)<=16: # пропускаем линии, близкие к горизонтальным
                continue
```

Проверяем, не слишком ли близко находится предыдущая линия (потому что одна граница могла определиться сразу несколькими линиями). Если проверка дала отрицательный результат, сразу присваиваем значения координатам предыдущей линии и переходим к следующей итерации:

```
# если предыдущая линия расположена не слишком близко к текущей
> if ((abs(prevx1-x1)>= height/30)&(abs(prevx2-x2)>= height/30))|(k==0)):
    prevx1 = x1
    prevy2 = y2
    prevy1 = y1
    prevx2 = x2
```

В случае успешной проверки увеличиваем счетчик книжных границ и рисуем линию:

```
continue
# если предыдущая линия расположена не слишком близко к текущей
if ((abs(prevx1-x1)>= height/30)&(abs(prevx2-x2)>= height/30))|(k==0)):

    k+=1 # увеличиваем счетчик границ

    cv2.line(line_image,(x1,y1),(x2,y2),(255,0,0),3) # рисуем синюю линию
```

Если это была не первая граница первой книги, то увеличиваем счетчик книг, меняем местами координаты верхнего и нижнего концов предыдущей линии, если это необходимо (координата с индексом «1» должна быть верхней). То же самое делаем и с текущей линией. Рисуем предыдущую линию:

```
cv2.line(line_image,(x1,y1),(x2,y2),(255,0,0),3) # рисуем синюю линию

if (k>1):      # если не первая граница первой книги

    count +=1 # увеличиваем счетчик книг

# меняем местами координаты по оси y, если нужно
if (prevy1> prevy2) :
    z=prevy1
    prevy1=prevy2
    prevy2=z
if (y1> y2) :
    z=y1
    y1=y2
    y2=z

# рисуем предыдущую линию синим цветом
cv2.line(line_image,(prevx1,prevy1),(prevx2,prevy2),(255,0,0),3)
```

Вырезаем прямоугольный фрагмент книги с помощью найденных линий. Координаты x выбираем так, чтобы как можно меньше захватить другие книги: выбираем самую правую координату x левой границы книги и самую левую координату x правой границы:

```
max(prevx1,prevx2):min(x1,x2)
```

В связи с тем, что одни книги выше других, в качестве верхней координаты у берется максимальная из координат у верхних концов левой и правой линии, чтобы не захватить лишнее пространство над книгой. Полагаем, что нижние концы книг стоят на полке примерно на одном уровне, поэтому в качестве нижней координаты у берем максимальную из координат у нижних концов границ книги:

```
max(prevy1,y1):max(prevy2,y2)
```

Рисуем диагональ фрагмента:

```
# рисуем диагональ фрагмента книги зеленым цветом
cv2.line(line_image,(max(prevx1,prevx2),max(prevy1,y1)),(min(x1,x2),max(prevy2,y2)),(0,255,0),2)
```

Ниже представлены примеры выделенных таким образом фрагментов (синим цветом показаны границы книги, найденные методом Хафа; зеленым – диагональ фрагмента). Понятно, что чем ровнее стоит книга на полке, тем лучше будет результат (на втором примере белым цветом показаны границы фрагмента):



Выводим на экран фрагмент книги вместе с ее номером. Далее считаем средний цвет фрагмента (пока что в формате BGR) и закрашиваем им прямоугольник, полученный методом cv2.rectangle. Затем преобразуем полученный в формате BGR средний цвет в формат RGB – переворачиваем массив и выбираем только последние 3 элемента (первый отвечает за прозрачность, поэтому его опускаем). Выводим средний цвет со строкой RGB и прямоугольником:

```

# рисуем диагональ фрагмента книги зеленым цветом
cv2.line(line_image,(max(prevx1,prevx2),max(prevy1,y1)),(min(x1,x2),max(prevy2,y2)),(0,255,0),2)

print("\n---Книга №"+str(count)+"---\n")

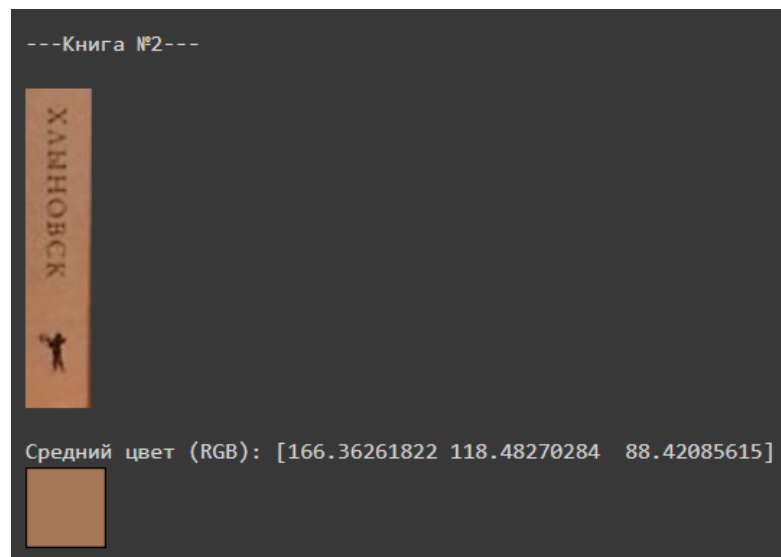
# выводим фрагмент на экран
fragment =image[max(prevy1,y1):max(prevy2,y2),max(prevx1,prevx2):min(x1,x2)]
cv2_imshow(fragment)

# считаем средний цвет фрагмента (пока bgr)
col = np.array(cv2.mean(fragment))
# рисуем квадрат и закрашиваем его средним цветом фрагмента
rect = np.zeros([50,50,3], np.uint8)
rect = cv2.rectangle(rect,(1,1),(48,48),(int(col[0]),int(col[1]),int(col[2])),-1)
# переворачиваем массив цвета в BGR и выбираем последние 3 составляющие - получаем RGB
rgb = np.flip(col)[1:4]

print('\nСредний цвет (RGB): '+str(rgb))
cv2_imshow(rect)

```

Пример вывода кода:



Создаем массив `err`, каждым элементом которого будет являться оценка принадлежности книги к соответствующей индексу коллекции – отклонение цвета фрагмента от среднего цвета коллекции из массива `colours`. Т. е. чем больше значение элемента `err`, тем меньше вероятность того, что книга принадлежит этой коллекции.

Элемент массива вычисляется как сумма модуля разности среднего цвета фрагмента (`rgb`) и соответствующего элемента массива `colours`, к которой прибавлена сумма модулей разностей разностей каналов (красный-зеленый, красный-синий, зеленый-синий):

```

print("\nАнализ (средняя ошибка):\n")

# массив ошибок
err = np.zeros(len(names))

for c in range(0, len(colours)):
    # разность цветов
    mean = abs(rgb - colours[c])
    # разница разностей каналов
    rg = abs((rgb[0] - rgb[1]) - (colours[c][0] - colours[c][1])) # red-green
    rb = abs((rgb[0] - rgb[2]) - (colours[c][0] - colours[c][2])) # red-blue
    gb = abs((rgb[1] - rgb[2]) - (colours[c][1] - colours[c][2])) # green-blue

    err[c] = np.mean(mean + (rg + rb + gb))

```

Выводим на экран ошибки для каждой коллекции и результат, который можно узнать по индексу наименьшей ошибки. Увеличиваем соответствующий счетчик в массиве статистики (то есть увеличиваем количество книг этой коллекции):

```

print(names[c]+' : '+str(err[c]))

print('\nРЕЗУЛЬТАТ: название коллекции - "'+(names[np.argmin(err)])+'"')

stat[np.argmin(err)] += 1

```

Пример вывода:

```

Анализ (средняя ошибка):

Librarium: 32.086314833449826
Великая поэзия: 62.6659697950768
Азбука Классика: 220.11257439479073
Попаданчество: 278.0416183190268
Всемирная литература: 220.47623439913025
Любовь. Интрига. Тайна: 187.9739275598549
Николай Стариков: 315.2239433641362

РЕЗУЛЬТАТ: название коллекции - "Librarium"

```

В этом случае наименьшая ошибка (32.08631..) соответствует коллекции «Librarium».

В конце программы печатаем количество книг и таблицу, в которой показано, сколько книг из каждой коллекции нашлось на полке:

```

print("\n-----\n")
print("СТАТИСТИКА:\n")
print("Найдено книг: "+ str(count))
print()
for i in range (0,len(names)):
    if i != 6:
        print("%22s"% (names[i]),end=' | ')
    else:
        print("%22s"% (names[i]),end='')
print()
for i in range (0,23*7+3):
    if (i!=0)&(i%23 == 0)&(i!=23*7):
        print("-|-",end='')
    elif i!=0:
        print("-",end='')
print()
for i in range (0,len(names)):
    if i != 6:
        print("%22d"% (stat[i]), end=' | ')
    else:
        print("%22d"% (stat[i]), end='')

```

Для справки выводим изображение с выделенными границами и диагоналями фрагментов:

```

print("\n\n* Результат детектирования корешков книг на полке:\n")
lines_edges = cv2.addWeighted(image, 0.5, line_image,1,0)
l_img = np.hstack([image, lines_edges])
cv2_imshow(l_img)

```

Полный пример работы представлен ниже:





---Книга №1---

Средний цвет (RGB): [151.12305854 107.35273219 79.7448909 ]

Анализ (средняя ошибка):

Librarium: 30.640957831467635  
 Великая поэзия: 46.07612713416987  
 Азбука Классика: 195.3035536793473  
 Попаданчество: 276.5962613170446  
 Всемирная литература: 201.45119051451948  
 Любовь. Интрига. Тайна: 163.1649068444114  
 Николай Стариков: 321.92099601234577

РЕЗУЛЬТАТ: название коллекции - "Librarium"

---Книга №2---

Средний цвет (RGB): [166.36261822 118.48270284 88.42085615]

Анализ (средняя ошибка):

Librarium: 32.086314833449826  
 Великая поэзия: 62.6659697950768  
 Азбука Классика: 220.11257439479073  
 Попаданчество: 278.0416183190268  
 Всемирная литература: 220.47623439913025  
 Любовь. Интрига. Тайна: 187.9739275598549  
 Николай Стариков: 315.2239433641362

РЕЗУЛЬТАТ: название коллекции - "Librarium"

---Книга №3---

Средний цвет (RGB): [180.58477079 134.35081111 105.9244532 ]

Анализ (средняя ошибка):

Librarium: 14.403821925121713  
 Великая поэзия: 75.25961154656078  
 Азбука Классика: 229.4143047443731  
 Попаданчество: 255.61411007151318  
 Всемирная литература: 220.9181609142449  
 Любовь. Интрига. Тайна: 197.2756579094372  
 Николай Стариков: 324.8990390156502

РЕЗУЛЬТАТ: название коллекции - "Librarium"

---Книга №4---

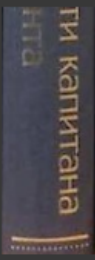
Средний цвет (RGB): [90.60844545 88.84583246 93.1396626 ]

Анализ (средняя ошибка):


Librarium: 189.7324929999414  
 Великая поэзия: 162.12363934597226  
 Азбука Классика: 62.7368279472087  
 Попаданчество: 151.37700042055937  
 Всемирная литература: 24.49502551780979  
 Любовь. Интрига. Тайна: 58.69698799453943  
 Николай Стариков: 424.4039177577965

РЕЗУЛЬТАТ: название коллекции - "Всемирная литература"

---Книга №5---



Средний цвет (RGB): [80.44683155 74.76031457 79.64719626]




Анализ (средняя ошибка):


Librarium: 195.65065416496805  
 Великая поэзия: 152.4429707032049  
 Азбука Классика: 60.5596923849529  
 Попаданчество: 171.8046744462898  
 Всемирная литература: 42.690240982334906  
 Любовь. Интрига. Тайна: 39.455417070549714  
 Николай Стариков: 410.7506529994378

РЕЗУЛЬТАТ: название коллекции - "Любовь. Интрига. Тайна"

---Книга №6---



Средний цвет (RGB): [83.92891484 75.54258242 81.63175366]




Анализ (средняя ошибка):


Librarium: 190.57263287927353  
 Великая поэзия: 148.687987685058  
 Азбука Классика: 67.52078086843711  
 Попаданчество: 175.12133580395297  
 Всемирная литература: 43.60232324099511  
 Любовь. Интрига. Тайна: 39.25995998359278  
 Николай Стариков: 405.11260278350125

РЕЗУЛЬТАТ: название коллекции - "Любовь. Интрига. Тайна"

---Книга №7---



Средний цвет (RGB): [231.31254529 226.00321558 224.54542572]




Анализ (средняя ошибка):


Librarium: 213.12797330801746  
 Великая поэзия: 254.58548031923536  
 Азбука Классика: 206.25267674656266  
 Попаданчество: 33.60690764643724  
 Всемирная литература: 172.13214868427167  
 Любовь. Интрига. Тайна: 176.52272991859437  
 Николай Стариков: 459.93016645473335

РЕЗУЛЬТАТ: название коллекции - "Попаданчество"

---Книга №8---



Средний цвет (RGB): [39.21161477 56.32234432 44.01884921]




Анализ (средняя ошибка):

Librarium: 238.63157162571224  
 Великая поэзия: 184.4051343260328  
 Азбука Классика: 25.225266776048045  
 Попаданчество: 219.222575565349  
 Всемирная литература: 92.84719996438747  
 Любовь. Интрига. Тайна: 68.54077039135635  
 Николай Стариков: 469.715750359305

РЕЗУЛЬТАТ: название коллекции - "Азбука Классика"

---Книга №9---



Средний цвет (RGB): [51.3243841 70.29213 51.91724519]

Анализ (средняя ошибка):

Librarium: 218.87584128277297  
 Великая поэзия: 169.75719643319707  
 Азбука Классика: 18.175115073839816  
 Попаданчество: 220.03837128377575  
 Всемирная литература: 93.6629956828142  
 Любовь. Интрига. Тайна: 65.10880072765933  
 Николай Стариков: 469.5873664327875

РЕЗУЛЬТАТ: название коллекции - "Азбука Классика"

---Книга №10---



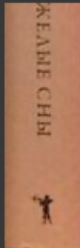
Средний цвет (RGB): [178.83269841 133.84790123 106.11710758]

Анализ (средняя ошибка):

Librarium: 9.698489509140515  
 Великая поэзия: 73.18104039302388  
 Азбука Классика: 224.8374085874259  
 Попаданчество: 252.41209915936045  
 Всемирная литература: 216.34126475729772  
 Любовь. Интрига. Тайна: 192.69876175249001  
 Николай Стариков: 327.8779696491459

РЕЗУЛЬТАТ: название коллекции - "Librarium"

---Книга №11---



Средний цвет (RGB): [182.79880508 126.90321135 96.6914115 ]

Анализ (средняя ошибка):

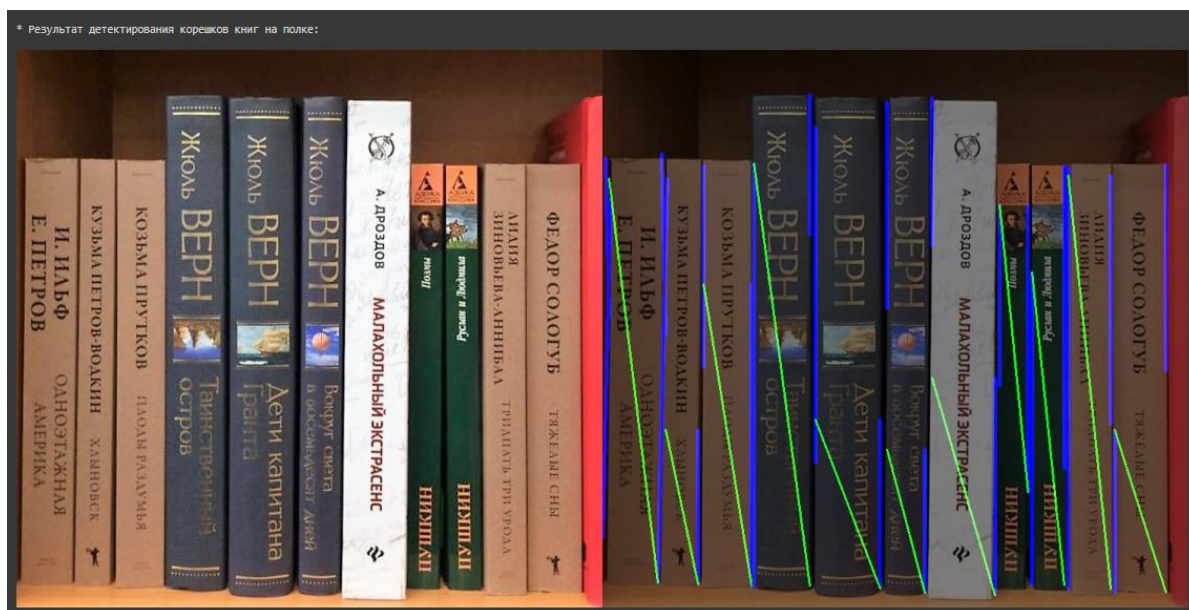
Librarium: 42.47520566887603  
 Великая поэзия: 86.37099490252365  
 Азбука Классика: 247.48625431758774  
 Попаданчество: 283.33046442268795  
 Всемирная литература: 242.33621075273837  
 Любовь. Интрига. Тайна: 215.34760748265185  
 Николай Стариков: 299.27754567502177

РЕЗУЛЬТАТ: название коллекции - "Librarium"

СТАТИСТИКА:

Найдено книг: 11

Librarium	Великая поэзия	Азбука Классика	Попаданчество	Всемирная литература	Любовь. Интрига. Тайна	Николай Стариков
5	0	2	1	1	2	0



В данном примере две книги (№5 и №6) классифицированы неправильно. Они принадлежат коллекции «Всемирная литература», но программа отнесла их к коллекции «Любовь. Интрига. Тайна», хотя их ошибки отличаются менее, чем на 5. Это произошло из-за сильной близости цветов выделенных фрагментов к темному цвету коллекции «Любовь. Интрига. Тайна».

Остальные 9 книг определены правильно.

## 2.2 Анализ точности

Из всех 218 книг на фотографиях программа правильно распознала 200 книг. Таким образом, точность:  $200/218 = 0.917431193$ .

Ниже представлена матрица неточностей:

	0	1	2	3	4	5	6	
40	1	0	0	0	0	0	0	0 = Librarium
1	33	0	0	0	0	0	0	1 = Великая поэзия
0	0	25	0	0	0	7	0	2 = Азбука Классика
0	0	0	37	0	0	0	0	3 = Попаданчество
0	0	0	0	29	9	0	0	4 = Всемирная литература
0	0	0	0	0	22	0	0	5 = Любовь. Интрига. Тайна
0	0	0	0	0	0	0	14	6 = Николай Стариков

Видно, что между собой были перепутаны следующие коллекции: «Librarium» и «Великая поэзия», «Азбука Классика» и «Любовь. Интрига. Тайна», «Всемирная литература» и «Любовь. Интрига. Тайна». Отсюда следует, что метод работает хуже

с книгами похожих цветов. Некоторое количество книг по ошибке отнеслось к коллекции «Любовь. Интрига. Тайна», имеющей самый темный средний цвет, потому что на более затемнённых фотографиях средний цвет книг становился темнее. Коллекции «Попаданчество» (белая) и «Николай Стариков» (красная) всегда определялись безошибочно и не перепутывались с другими коллекциями, так как имеют наиболее уникальные и яркие средние цвета.




## 3 Нейронная сеть

### 3.1 Описание работы программы

Импортируем библиотеку keras для работы с нейронными сетями:

```
from tensorflow import keras
```

В качестве основы для выборки будем снова использовать таблицу books data.xlsx. Данные для обучения расположены в столбцах A,B,C (см. п. 2.1), а для тестирования – в столбцах E, F, G:

	A	B	C	D	E	F	G
1			4 Всемирная Литература				2 Азбука Классика
2			1 Великая Поэзия				1 Великая Поэзия
3			3 Попаданчество				5 Любовь. Интрига. Тайна
4			5 Любовь. Интрига. Тайна				6 Николай Стариков
5			0 Librarium				5 Любовь. Интрига. Тайна
6			4 Всемирная Литература				5 Любовь. Интрига. Тайна
7			5 Любовь. Интрига. Тайна				0 Librarium
8			2 Азбука Классика				6 Николай Стариков
9			5 Любовь. Интрига. Тайна				4 Всемирная Литература
10			5 Любовь. Интрига. Тайна				4 Всемирная Литература
11			3 Попаданчество				1 Великая Поэзия
12			0 Librarium				6 Николай Стариков
13			5 Любовь. Интрига. Тайна				2 Азбука Классика
14			1 Великая Поэзия				5 Любовь. Интрига. Тайна
15			4 Всемирная Литература				1 Великая Поэзия

Для каждой книги сделано 13 записей для обучения и 4 для тестирования:

Создаем объекты для работы с таблицей, выбирая первый лист:

```
pxl_doc = openpyxl.load_workbook('/content/drive/MyDrive/bookshelves/books_data.xlsx')
sheet = pxl_doc['Лист1']

image_loader = SheetImageLoader(sheet)
```

Создаем массив названий коллекций:

```
names = ["Librarium", "Великая поэзия", "Азбука Классика", "Попаданчество", "Всемирная литература",
         "Любовь. Интрига. Тайна", "Николай Стариков"]
```

Структура нейронной сети будет иметь следующий вид:

- *входной слой*:  $32 \times 32 \times 3$  нейрона, задающих цветное (rgb) изображение размером  $32 \times 32$ , 3 нейрона для компонент среднего цвета этого изображения, 3 нейрона для разностей каналов (красный-зеленый, красный-синий, зеленый-синий). Всего:  $32 \times 32 \times 3 + 3 + 3 = 3078$  нейронов.

- *скрытый слой*: 1024 нейрона (их число выбрано эмпирически), функция активации – сигмоидальная.
- *выходной слой*: 7 нейронов (для каждой из семи коллекций), функция активации – softmax (возвращает вероятности принадлежности книги к той или иной коллекции).

Создаем пустые списки для входов и заполненные нулями массивы длины 7 для соответствующих коллекций («правильных ответов»):

```
n_train = 91 # кол-во записей обучающего набора
n_test = 28 # кол-во записей набора для тестирования

xtrain = [] # входы обучающего набора
xtest = [] # входы набора для тестирования

ytrain = np.zeros( (n_train,7)) # выходы обучающего набора
ytest = np.zeros( (n_test,7)) # выходы набора для тестирования
```

Организуем цикл по строкам таблицы. На каждой итерации получаем изображение для обучения, находим средний цвет в формате RGB и разности каналов. Объединяем в один массив:

```
for i in range(1, n_train+1):

    img = np.array(image_loader.get('A'+str(i))) # изображение
    rgb = np.array((cv2.mean(img))[:3]) # средний цвет
    rg = np.array(abs(rgb[0]-rgb[1])) # красный - зеленый
    rb = np.array(abs(rgb[0]-rgb[2])) # красный - синий
    gb = np.array(abs(rgb[1]-rgb[2])) # зеленый - синий

    # объединяем в один массив
    vec = (np.concatenate((rgb, rg, rb, gb),axis = None))
```

Объединяем результат и преобразованное к массиву изображение (с помощью метода `flatten()`), присоединяем к обучающей выборке:

```
xtrain.append(np.append((np.array(img)).flatten(), vec))
```

Записываем единицу в массив «правильных выходов» на место, соответствующее идентификатору из столбца B (индекс названия «правильной коллекции»):

```
ytrain[i-1][int((sheet['B'+str(i)]).value)]=1
```



Если в строке ещё есть тестовые данные, то выполняем аналогичные действия для xtest и ytest:

```
ytrain[i-1][int((sheet['B'+str(i)]).value)]=1

# аналогично, если в строке еще есть тестовая выборка
if i< n_test +1:
    img = np.array(image_loader.get('E'+str(i)))
    rgb = np.array((cv2.mean(img))[:3])
    rg = np.array(abs(rgb[0]-rgb[1]))
    rb = np.array(abs(rgb[0]-rgb[2]))
    gb = np.array(abs(rgb[1]-rgb[2]))

    vec = (np.concatenate((rgb, rg, rb, gb),axis = None))

    xtest.append(np.append((np.array(img)).flatten(), vec))

    ytest[i-1][int((sheet['F'+str(i)]).value)]=1
```

Преобразуем списки в массивы numpy и делим их элементы на 255 для нормализации (теперь все значения находятся в интервале [0, 1]):

```
xtrain = np.array(xtrain)
xtest = np.array(xtest)

# # нормализация
xtrain =xtrain/255
xtest = xtest/255
```

С помощью библиотеки keras строим полносвязную нейронную сеть в соответствии с вышеописанной структурой:

```
# последовательная модель нейронной сети
model = keras.Sequential()
# входной слой
model.add(keras.layers.Input(shape=((32*32*3+6),)))
# скрытый слой
# (Dense обеспечивает полносвязность)
model.add(keras.layers.Dense(1024, activation='sigmoid'))
# выходной слой
model.add(keras.layers.Dense(7, activation='softmax'))
```

Компилируем. В качестве оптимизатора используем Adam, функции потерь – категориальную кросс-энтропию (показывает, насколько близка вероятность предсказания к соответствующему истинному значению) и ассигасу (отношение

правильных ответов к общему числу ответов: т. к. в данной задаче важнее всего именно количество правильно определённых книг) как метрику.

```
model.compile(optimizer=keras.optimizers.Adam(),
               loss=keras.losses.CategoricalCrossentropy(),
               metrics=['accuracy'])
```

Обучаем модель, используя метод fit. Указываем количество эпох равным 30:

```
# обучение
model.fit(xtrain, ytrain, epochs = 30)
```

Запускаем процесс обучения:

```
Epoch 1/30
3/3 [=====] - 1s 73ms/step - loss: 2.8163 - accuracy: 0.2088
Epoch 2/30
3/3 [=====] - 0s 39ms/step - loss: 2.1772 - accuracy: 0.3626
Epoch 3/30
3/3 [=====] - 0s 58ms/step - loss: 1.2405 - accuracy: 0.4945
Epoch 4/30
3/3 [=====] - 0s 46ms/step - loss: 0.9272 - accuracy: 0.5934
Epoch 5/30
3/3 [=====] - 0s 25ms/step - loss: 0.7658 - accuracy: 0.7582
Epoch 6/30
```

...

```
Epoch 28/30
3/3 [=====] - 0s 25ms/step - loss: 0.0493 - accuracy: 1.0000
Epoch 29/30
3/3 [=====] - 0s 26ms/step - loss: 0.0463 - accuracy: 1.0000
Epoch 30/30
3/3 [=====] - 0s 24ms/step - loss: 0.0439 - accuracy: 1.0000
```

За все время обучения потеря понизилась со значения 2.8163 до 0.0439, а точность повысилась с 0.2088 до 1. В связи с тем, что начальные веса генерируются случайным образом, результат обучения каждый раз будет разным, но для данной задачи это отличие не является существенным.

Проверим, как отработает модель на тестовых данных:

```
# проверка на тестовых данных
model.evaluate(xtest, ytest)
```

Результаты: потеря = 0.1873, точность = 0.9286:

```
TEST
1/1 [=====] - 0s 134ms/step - loss: 0.1873 - accuracy: 0.9286
[0.18731246888637543, 0.9285714030265808]
```

Переходим к использованию этой нейронной сети в классификации книг. Программа во многом повторяет код из п. 2. Теперь после выделения фрагмента книги, необходимо преобразовать его размер к 32×32 и перевести в формат RGB, вычислить его средний цвет и разности каналов среднего цвета – как это было сделано при подготовке нейросети:

```
img = cv2.resize(fragment,(32,32)) # меняем размер

img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB) # в RGB

rgb = np.array((cv2.mean(img))[:3])
rg = np.array(abs(rgb[0]-rgb[1]))
rb = np.array(abs(rgb[0]-rgb[2]))
gb = np.array(abs(rgb[1]-rgb[2]))

vec = (np.concatenate((rgb, rg, rb, gb),axis = None))
```

Используем модель нейронной сети для определения коллекции книги: для этого соединяем полученный ранее массив и преобразованное в массив изображение и подаём результат в метод predict. В цикле печатаем результаты предсказания (вероятности принадлежности книги каждой коллекции). Определяем итоговую коллекцию по максимальной вероятности.

```
print("\nАнализ:")
res = model.predict(np.expand_dims((np.append((np.array(img)).flatten(), vec))/255,axis = 0))

for i in range (0,len(names)):
    print(names[i]+' : '+str(res[0,i]))

print('\nРЕЗУЛЬТАТ: название коллекции - ''+(names[np.argmax(res)]+'''')
```

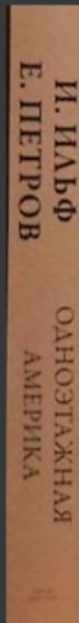
В примере ниже максимальная вероятность (0.9169579) соответствует коллекции «Librarium»:



Полный пример работы кода представлен ниже:



---Книга №1---



Средний цвет (RGB): [151.12305854 107.35273219 79.7448909 ]



**Анализ:**

```
1/1 [=====] - 0s 158ms/step
```

Librarium: 0.8356915

Великая поэзия: 0.14124149

Азбука Классика: 0.005167119

Попаданчество: 0.0013169922

Всемирная литература: 0.0018290869

Любовь. Интрига. Тайна: 0.008894133

Николай Стариков: 0.005859666

РЕЗУЛЬТАТ: название коллекции - "Librarium"

---Книга №2---



Средний цвет (RGB): [166.36261822 118.48270284 88.42085615]



**Анализ:**

```
1/1 [=====] - 0s 25ms/step
```

Librarium: 0.9169579

Великая поэзия: 0.068810314

Азбука Классика: 0.00074662943

Азбука Класика: 0.00074662  
Попаданчество: 0.0013454186

Всемирная литература: 0.0029016759

Любовь. Интрига. Тайна: 0.0028275147

Любовь. Интрига. Тайна: 0.002  
Николай Стариков: 0.006410601

РЕЗУЛЬТАТ: название коллекции - "Librarium"

---Книга №3---




Средний цвет (RGB): [180.58477079 134.35081111 105.9244532 ]




Анализ:  
 1/1 [=====] - 0s 24ms/step  
 Librarium: 0.96809196  
 Великая поэзия: 0.019681968  
 Азбука Классика: 0.0006452497  
 Попаданчество: 0.003900618  
 Всемирная литература: 0.0041734213  
 Любовь. Интрига. Тайна: 0.001249348  
 Николай Стариков: 0.0022574225

РЕЗУЛЬТАТ: название коллекции - "Librarium"

---Книга №4---



Средний цвет (RGB): [90.60844545 88.84583246 93.1396626 ]



Анализ:  
 1/1 [=====] - 0s 27ms/step  
 Librarium: 0.0064835865  
 Великая поэзия: 0.003215486  
 Азбука Классика: 0.01504854  
 Попаданчество: 0.0064380197  
 Всемирная литература: 0.9321681  
 Любовь. Интрига. Тайна: 0.036626182  
 Николай Стариков: 2.0073217e-05

РЕЗУЛЬТАТ: название коллекции - "Всемирная литература"

---Книга №5---



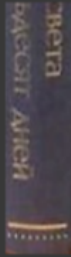
Средний цвет (RGB): [80.44683155 74.76031457 79.64719626]




Анализ:  
 1/1 [=====] - 0s 31ms/step  
 Librarium: 0.008063551  
 Великая поэзия: 0.007410409  
 Азбука Классика: 0.01300708  
 Попаданчество: 0.005824812  
 Всемирная литература: 0.9460322  
 Любовь. Интрига. Тайна: 0.019593747  
 Николай Стариков: 6.814144e-05

РЕЗУЛЬТАТ: название коллекции - "Всемирная литература"

---Книга №6---



Средний цвет (RGB): [83.92891484 75.54258242 81.63175366]



Анализ:  
 1/1 [=====] - 0s 28ms/step  
 Librarium: 0.03247739  
 Великая поэзия: 0.016607098  
 Азбука Классика: 0.025247725  
 Попаданчество: 0.03388667  
 Всемирная литература: 0.8673701  
 Любовь. Интрига. Тайна: 0.024240848  
 Николай Стариков: 0.0001700604

РЕЗУЛЬТАТ: название коллекции - "Всемирная литература"

---Книга №7---



Средний цвет (RGB): [231.31254529 226.00321558 224.54542572]



Анализ:  
 1/1 [=====] - 0s 25ms/step  
 Librarium: 0.010289076  
 Великая поэзия: 0.00011316159  
 Азбука Классика: 0.0005300928  
 Попаданчество: 0.93947583  
 Всемирная литература: 0.049475063  
 Любовь. Интрига. Тайна: 0.0001146709  
 Николай Стариков: 2.1470858e-06

РЕЗУЛЬТАТ: название коллекции - "Попаданчество"

---Книга №8---



Средний цвет (RGB): [39.21161477 56.32234432 44.01884921]




Анализ:  
 1/1 [=====] - 0s 22ms/step  
 Librarium: 0.0010180582  
 Великая поэзия: 0.003933523  
 Азбука Классика: 0.877884  
 Попаданчество: 0.0005387711  
 Всемирная литература: 0.018128876  
 Любовь. Интрига. Тайна: 0.098491855  
 Николай Стариков: 4.8959805e-06


РЕЗУЛЬТАТ: название коллекции - "Азбука Классика"



---Книга №9---



Средний цвет (RGB): [51.3243841 70.29213 51.91724519]



Анализ:  
 1/1 [=====] - 0s 23ms/step  
 Librarium: 0.00039333897  
 Великая поэзия: 0.018881954  
 Азбука Классика: 0.87927824  
 Попаданчество: 0.00027926965  
 Всемирная литература: 0.026924193  
 Любовь. Интрига. Тайна: 0.07424067  
 Николай Стариков: 2.2692402e-06

РЕЗУЛЬТАТ: название коллекции - "Азбука Классика"

---Книга №10---



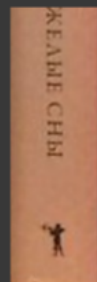
Средний цвет (RGB): [178.83269841 133.84790123 106.11710758]




Анализ:  
 1/1 [=====] - 0s 24ms/step  
 Librarium: 0.9575465  
 Великая поэзия: 0.031698175  
 Азбука Классика: 0.0020238727  
 Попаданчество: 0.002762975  
 Всемирная литература: 0.0027614508  
 Любовь. Интрига. Тайна: 0.0015837299  
 Николай Стариков: 0.0016232664

РЕЗУЛЬТАТ: название коллекции - "Librarium"

---Книга №11---



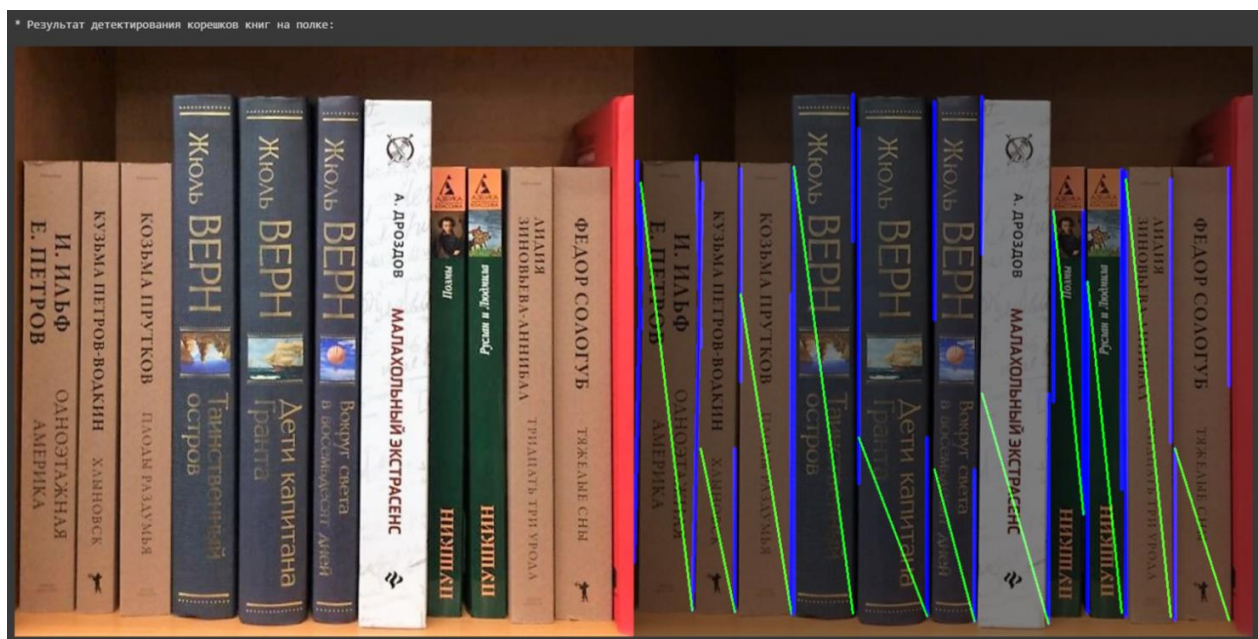
Средний цвет (RGB): [182.79880508 126.90321135 96.6914115 ]



Анализ:  
 1/1 [=====] - 0s 22ms/step  
 Librarium: 0.955831  
 Великая поэзия: 0.03541971  
 Азбука Классика: 0.00047140996  
 Попаданчество: 0.000846328  
 Всемирная литература: 0.0011115116  
 Любовь. Интрига. Тайна: 0.0012695997  
 Николай Стариков: 0.005050485

РЕЗУЛЬТАТ: название коллекции - "Librarium"

СТАТИСТИКА:						
Найдено книг: 11						
Librarium	Великая поэзия	Азбука Классика	Попаданчество	Всемирная литература	Любовь. Интрига. Тайна	Николай Стариков
5	0	2	1	3	0	0



В данном случае все книги классифицированы верно.

### 3.2 Анализ точности

Из всех 218 книг на фотографиях программа правильно распознала 207 книг.

Таким образом, точность:  $207/218 = 0.949541284$ .

Ниже представлена матрица неточностей:

	0	1	2	3	4	5	6	
41	0	0	0	0	0	0	0	0 = Librarium
0	34	0	0	0	0	0	0	1 = Великая поэзия
0	0	29	0	2	1	0	0	2 = Азбука Классика
0	0	0	36	1	0	0	0	3 = Попаданчество
0	0	1	0	37	0	0	0	4 = Всемирная литература
0	3	0	0	3	16	0	0	5 = Любовь. Интрига. Тайна
0	0	0	0	0	0	0	14	6 = Николай Стариков

Книги из коллекций «Librarium», «Великая поэзия» и «Николай Стариков» были классифицированы безошибочно. Хуже всего программа показала себя на книгах коллекции «Любовь. Интрига. Тайна». Книги этой коллекции имеют черный цвет. На сильно светлых изображениях нейросеть относил их к другим более менее «темным» коллекциям – «Азбука Классика» и «Всемирная литература».

В целом, метод, использующий нейронную сеть, оказался немного эффективнее метода, использующего только `opencv`, правильно определив на 7 книг больше.

## 4 MatchTemplate

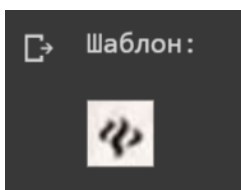
Если на корешках книг определённой коллекции есть какой-либо четкий узор или значок, то, при условии, что все фотографии книжных полок примерно одного размера, можно воспользоваться методом `cv2.matchTemplate` для поиска шаблонов на изображении.

Например, на книгах коллекции «Попаданчество» имеется характерный узор:



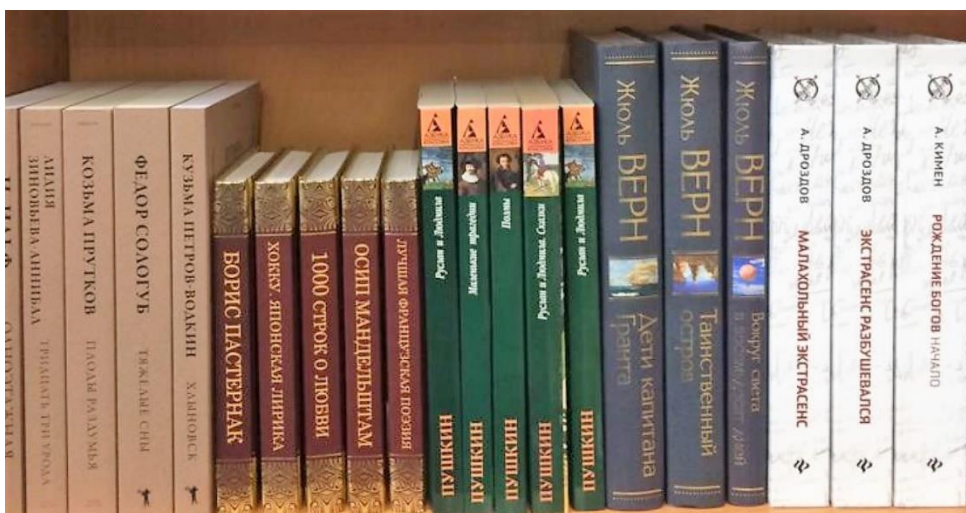
Используем его в качестве шаблона:

```
print("Шаблон:\n");  
  
template = cv2.imread("/content/drive/MyDrive/bookshelves/template.jpg")  
  
cv2_imshow(template)
```



Загружаем изображение, на котором хотим найти такие книги:

```
print("\nИзображение с книгами:\n");  
  
templ_img= cv2.imread("/content/drive/MyDrive/bookshelves/bs3.JPG")  
  
cv2_imshow(templ_img)
```



Переводим шаблон и фотографию книжной полки в градации серого и используем метод `matchTemplate`. Указываем пороговое значение (т.к. на фото может быть больше одной книги):

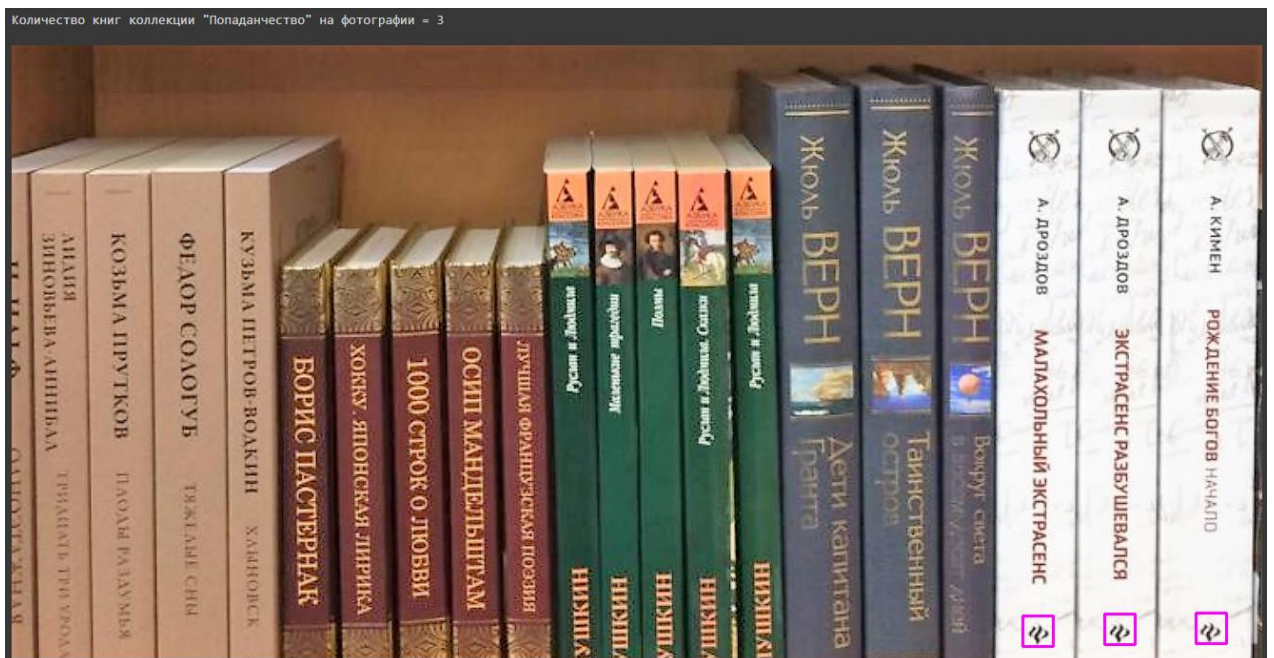
```
# размеры шаблона
w, h = template_gray.shape[:2]
# находим шаблон на изображении
result = cv2.matchTemplate(image_gray, template_gray, cv2.TM_CCOEFF_NORMED)
threshold = 0.7 # пороговое значение
loc = np.where(result >= threshold)
```

Сортируем найденные объекты по сумме координат x и y и организуем цикл, в котором обводим объекты прямоугольниками и считаем их количество, пропуская лишние найденные объекты (метод может несколько раз найти один и тот же объект – поэтому считаем только те объекты, которые не сливаются в один)

```
count = 0 # количество найденных шаблонов
prevx = 0
prevy = 0
flag = 0
# обводим найденные объекты прямоугольниками
for pt in sorted(zip(*loc[:, :2]), key=lambda p: p[0]+p[1]):
    if(flag==0 or (abs(pt[0]-prev[0])>2 and abs(pt[1]-prev[1])>2 )) :
        count += 1
        cv2.rectangle(templ_img, pt, (pt[0] + w, pt[1] + h), (255, 0, 255), 2)

    prev=pt
    flag = 1
# показываем результат
print('\nКоличество книг коллекции "Попаданчество" на фотографии = '+str(count)+'\n')
cv2.imshow(templ_img)
```

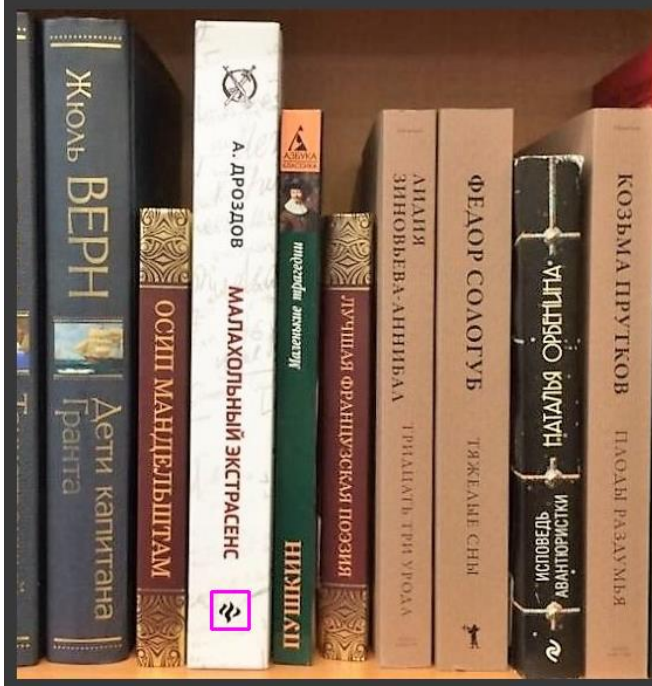
Результат:



Ещё несколько примеров работы кода:



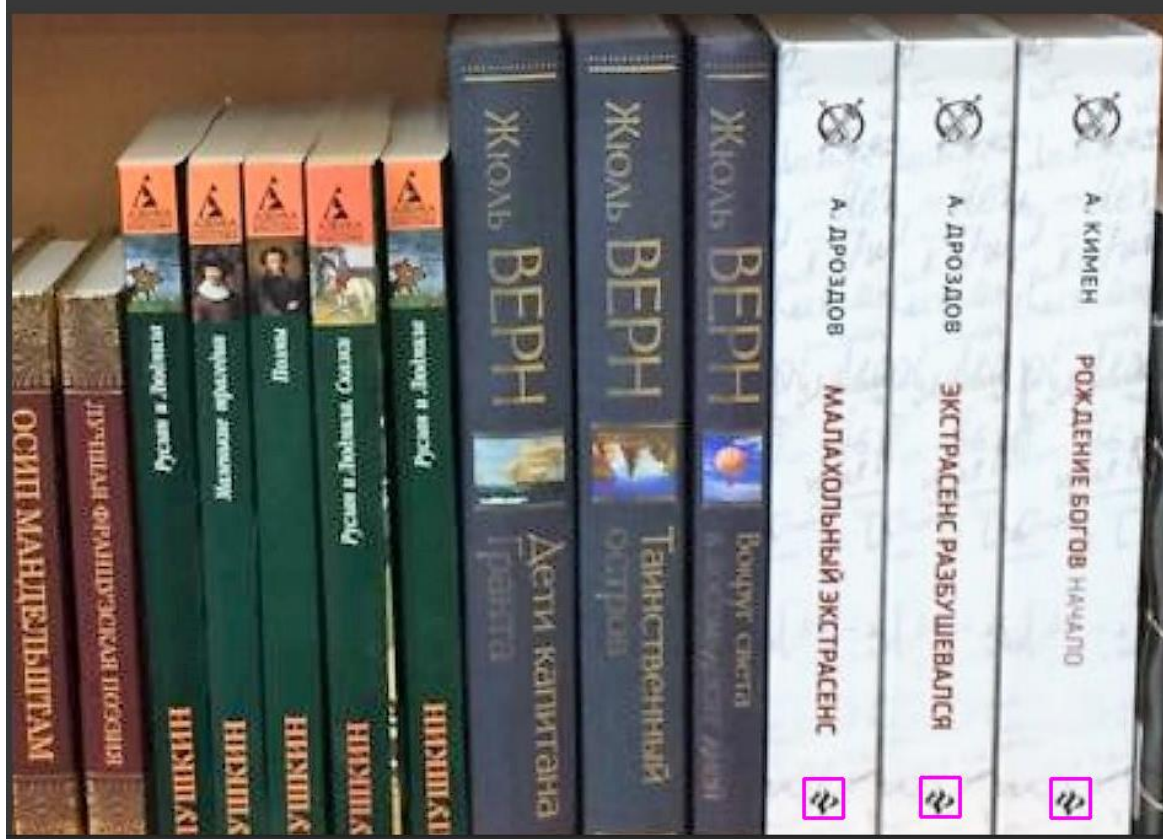
Количество книг коллекции "Попаданчество" на фотографии = 1



Количество книг коллекции "Попаданчество" на фотографии = 2



Количество книг коллекции "Попаданчество" на фотографии = 3



## 5 Выводы

Сравним точности классификации книг, полученные разными методами:

Метод, использующий только `openCV`:

- Точность: 0.917431193
- Верно определенных книг: 200

Метод с добавлением нейронной сети:

- Точность: 0.949541284
- Верно определенных книг: 207

Таким образом, метод, использующий нейронную сеть, сработал точнее на  $0.949541284 - 0.917431193 = 0.032110091$ , правильно определив на  $207 - 200 = 7$  книг больше.

Сопоставим матрицы неточностей:

Метод, использующий только `openCV`:

0	1	2	3	4	5	6	
40	1	0	0	0	0	0	0 = Librarium
1	33	0	0	0	0	0	1 = Великая поэзия
0	0	25	0	0	7	0	2 = Азбука Классика
0	0	0	37	0	0	0	3 = Попаданчество
0	0	0	0	29	9	0	4 = Всемирная литература
0	0	0	0	0	22	0	5 = Любовь. Интрига. Тайна
0	0	0	0	0	0	14	6 = Николай Стариков

Метод с добавлением нейронной сети:

0	1	2	3	4	5	6	
41	0	0	0	0	0	0	0 = Librarium
0	34	0	0	0	0	0	1 = Великая поэзия
0	0	29	0	2	1	0	2 = Азбука Классика
0	0	0	36	1	0	0	3 = Попаданчество
0	0	1	0	37	0	0	4 = Всемирная литература
0	3	0	0	3	16	0	5 = Любовь. Интрига. Тайна
0	0	0	0	0	0	14	6 = Николай Стариков

По матрицам видно, что оба метода хуже всего классифицируют близкие по цвету книги, особенно темных оттенков. Первый метод чаще относит темные книги к коллекции «Любовь. Интрига. Тайна», которая имеет самый темный цвет, в то время



как второй метод, наоборот, чаще относит книги этой коллекции к более «светлым» коллекциям.

В случае, когда на корешке книги определенной коллекции есть какой-либо узнаваемый узор, отличающий ее от других коллекций, удобно использовать метод `matchTemplate`, используя такой узор в качестве шаблона. Если анализируемые фотографии имеют приблизительно одинаковый размер, с помощью этого метода можно весьма точно выделить эти узоры только на нужных книгах.