Валентина Старикова

А-05-18 | в-21

Отчет по лабораторной работе 8

часть 1

1. Номер варианта и задание.

**Задача:**

Создание набора данных, состоящего из зашумленных изображений различных фигур(кривых), и его проверка на соответствие требованиям (неповторяемость, широта представления и тд ).

21. Правильный пятиугольник и правильный пятиугольник без одной стороны. Варьируются радиус окружности, в которую вписан пятиугольник, и угол поворота пятиугольника относительно своего центра.

22. Локон Аньези x = 2actgt; y = 2asin2t (0 < t < π)  
и декартов лист x = a(t2 - 1) / (3t2 + 1); y = at(t2 - 1) / (3t2 + 1) (-∞ < t < ∞). Варьируется a.

23. Один период y = acosec(bx) и половина периода этой функции с y > 0 или y < 0. Варьируются коэффициенты a и b и их знаки. Выбор между y > 0 и y < 0 выполняется случайным образом.

1. Диапазоны изменения варьируемых параметров.

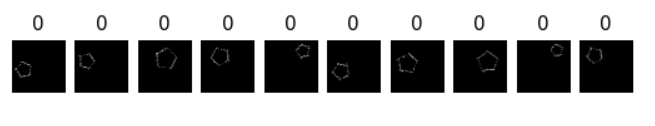
|  |  |  |
| --- | --- | --- |
| Имя параметра | min | max |
|  | **Пятиугольник** | |
| **Длина стороны** | 10 | 20 |
| **Угол поворота** | -90 | 90 |
|  | **Пятиугольник без строны** | |
| **Длина стороны** | 10 | 20 |
| **Угол поворота** | -90 | 90 |
| **Исчезнувшая стороны** | 1 | 5 |
|  | **Декартов Лист** | |
| **Коэфициент а** | -20 | 21 |
|  | **Локон Аньези** | |
| **Коэфициент а** | 1 | 20 |
|  | **Арксинус** | |
| **Коэфициент а** | -10 | 10 |
| **Коэфициент b** | 0.1 | 1 |
|  | **Арксинус 1/2** | |
| **Знак** | - | + |
| **Коэфициент а** | -10 | 10 |
| **Коэфициент b** | 0.1 | 1 |

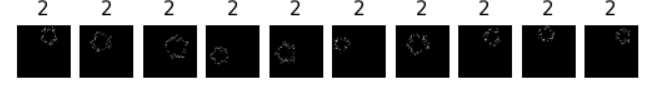
1. Описание приемов, использованных при внесении шума в генерируемые изображения.

С помощью функции random , генерирую значения x\_noise, y\_noise, и прибавляю их к каждой вычисленной точке фигуры.В итоге, линии фигуры получаются разной ширины.

1. Примеры сгенерированных изображений (по 10 в каждом классе).

Пятиугольник

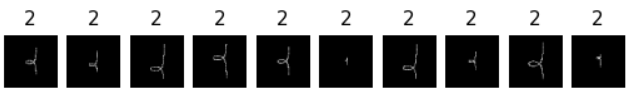
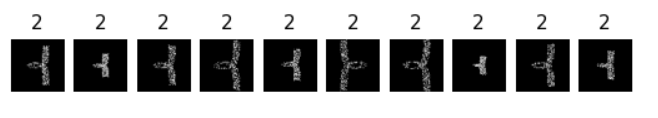




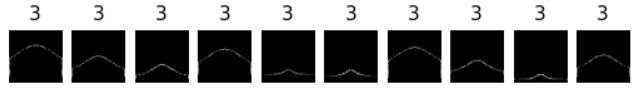
Пятиугольник без одной стороны



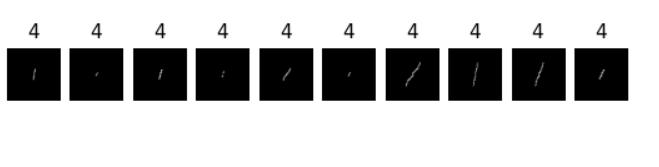
Декартов Лист

****

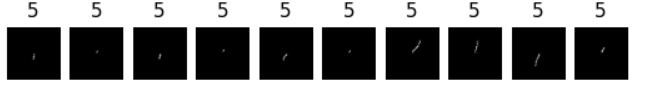
Локон Аньези



Арксинус



Арксинус ½



1. **Алгоритм проверки класса на неповторяемость.**

def iss(data): #функция нахождения минимального иср

m = 10000000

for i in range(len(data)):

im1 = data[i].reshape(w\*h)

for j in range(i + 1, len(data)):

im2 = data[j].reshape(w\*h)

iss = (1-structural\_similarity(im1, im2))/2

if iss < m:

m = iss

return m

def check\_sim(data,iss):# функция проверки на неповторяемость, подается iss=иср мин

n = len(data)

cnt = 0

for i in range(n):

im1 = data[i].reshape(w\*h)

for j in range(i + 1, n):

im2 = data[j].reshape(w\*h)

d = (1-structural\_similarity(im1, im2))/2

if d > iss:

return False

1. **Результаты проверки классов на неповторяемость (в виде таблицы).**

В таблицу занесены значения ИСР\_мин, количество аналогов

|  |  |  |
| --- | --- | --- |
| **Обучающее множество** | **Тестовое**  **множество** | **Все**  **множество** |
| Декартов Лист | | |
| 0.00014725770639587946 | 0.005625730001409386 | 0.0012451188967982363 |
| 2 | 10 |  |
| Пятиугольник | | |
| 0.026673992275908685 | 0.03481494609818568 | 0.019033919331206073 |
| 13 | 2 |  |
| Пятиугольник без стороны | | |
| 0.022363701665718855 | 0.02260816316009734 | 0.017338973605607266 |
| 14 | 1 |  |
| Локон Аньези | | |
| 0.00011710645365735813 | 0.021994429732393045 | 0.0013205347890931085 |
| 16 | 9 |  |
| Арксинус | | |
| 0.00002094014434878 | 0.0002992360285490048 | 0.0000102345455323428 |
| 13 | 1 |  |
| Арксинус 1/2 | | |
| 0.002703446739201573 | 0.005335640561611205 | 0.0010256387621738121 |
| 4 | 12 |  |

1. Обоснование значения ИСР\_мин (ИСР – индекс структурного различия), применяемого при поиске похожих изображений.

При поиске ИСР\_мин я каждый найденный минимум ИСР заношу в массив, а потом срезаю этот массив и использую последнюю 1\3 этого массива.Нахожу его среднее арифметическое этой 1\3, которое и будет являться ИСР\_мин.

1. Программа генерации, визуализации и проверки классов на неповторяемость.

Для каждой фигуры отдельный скрипт

**Программы генерации**

**Пятиугольник**

import numpy as np

import math, time

import matplotlib.pyplot as plt

from PIL import Image # Для поворота изображения

from skimage.metrics import structural\_similarity

#

np.random.seed(348)

full = not True # Полный прямоугольник, если True

cls = 0 if full else 1

show = not True

show\_test = True

fn\_train = 'dataTrain.bin'

fn\_train\_labels = ')labelsTrain.bin'

fn\_test = 'dataTest.bin'

fn\_test\_labels = 'labelsTest.bin'

n\_train = 600 # Число рисунков для обучения

n\_test = 700 # Число тестовых рисунков

clr\_mim, clr\_max = 75, 255 # Диапазон оттенков серого цвета

w, h = 64, 64 # Ширина и высота рисунка

w2 = w / 2

border = 4 # Граница

#

def iss(data):

m = 10000000

for i in range(len(data)):

im1 = data[i].reshape(w\*h)

for j in range(i + 1, len(data)):

im2 = data[j].reshape(w\*h)

iss = (1-structural\_similarity(im1, im2))/2

if iss < m:

m = iss

return m

def check\_sim(data,iss):

n = len(data)

cnt = 0

for i in range(n):

im1 = data[i].reshape(w\*h)

for j in range(i + 1, n):

im2 = data[j].reshape(w\*h)

d = (1-structural\_similarity(im1, im2))/2

if d > iss:

return False

def line(arrPic, x0, y0, x1, y1, hor = True):

for i in range (int(x0),int(x1)+1):

y\_noise = np.random.uniform(-4,4)

y = int((y1+y\_noise - y0)/(x1 - x0)\*(i - x0) + y0 + y\_noise)

arrPic[i, y] = np.random.randint(100, 255)

def pent(arrPic):

a = np.random.randint(10, 20)

coords=[[0.6\*a,1.4\*a], [1.3\*a,0.9\*a], [2\*a,1.4\*a], [1.7\*a,2.2\*a ], [0.9\*a,2.2\*a]]

if full:

idx = range(5)

else:

mis = np.random.randint(5) # Номер отсутствующей стороны

idx = [i for i in range(5) if i != mis]

for i in idx:

if i == 0: line(arrPic, coords[0][0], coords[0][1], coords[1][0], coords[1][1])

if i == 1: line(arrPic, coords[1][0], coords[1][1], coords[2][0], coords[2][1])

if i == 2: line(arrPic, coords[3][0], coords[3][1], coords[2][0], coords[2][1])

if i == 3: line(arrPic, coords[4][0], coords[4][1],coords[3][0], coords[3][1])

if i == 4: line(arrPic, coords[0][0], coords[0][1], coords[4][0], coords[4][1])

#

def prepareData(n, fn, fn2):

file = open(fn, 'wb')

file2 = open(fn2, 'wb')

xs0 = border - w2 + 1

xe = w2 - border - 1

dx = 0.1

for i in range(n):

if full: # Прямоугольник

xs = xs0

label = cls

else: # Прямоугольник без одной стороны

xs = xs0

label = cls

arrPic = np.zeros((w, h), dtype = np.uint8)

pent(arrPic)

ang = np.random.randint(-90, 90)

arrPic = rot\_img(ang, arrPic)

file.write(arrPic)

file2.write(np.uint8(label))

file.close()

file2.close()

#

def load\_data(fn, fn2):

with open(fn, 'rb') as read\_binary:

data = np.fromfile(read\_binary, dtype = np.uint8)

with open(fn2, 'rb') as read\_binary:

labels = np.fromfile(read\_binary, dtype = np.uint8)

return data, labels

#

def rot\_img(ang, img\_array):

# Приводим данные к типу uint8

img\_array = np.array(img\_array, dtype = 'uint8')

# Формируем изображение по массиву img\_array

img = Image.fromarray(img\_array, 'L')

# Поворот изображения на угол ang против часовой стрелки

img = img.rotate(ang)

# Переводим изображение в массив

ix = img.size[0]

iy = img.size[1]

img\_array\_rot = np.array(img.getdata(), dtype = 'uint8').reshape(iy, ix)

return img\_array\_rot

if not show:

t0 = time.time()

print('Поехали')

one\_class(n\_train, fn\_train, fn\_train\_labels)

one\_class(n\_test, fn\_test, fn\_test\_labels)

print('Потрачено времени:', round(time.time() - t0, 3))

def plotData(data, ttl, cls):

plt.figure(ttl)

k = 0

for i in range(10):

j = np.random.randint(data.shape[0])

k += 1

plt.subplot(1, 10, k)

plt.imshow(data[i], cmap = 'gray')

plt.title(cls, fontsize = 11)

plt.axis('off')

plt.subplots\_adjust(hspace = -0.1) # wspace

plt.show()

if show\_test:

test\_data, \_ = load\_data(fn\_test, fn\_test\_labels)

test\_data = test\_data.reshape(n\_test, w, h)

min\_iss\_test=iss(test\_data)

print('Минимальный индекс структурного различия в проверочном множестве',min\_iss\_test)

print('Проверка на неповторяемость', check\_sim(test\_data, min\_iss\_test ))

train\_data, \_ = load\_data(fn\_train, fn\_train\_labels)

train\_data = train\_data.reshape(n\_train, w, h)

min\_iss\_train=iss(train\_data)

print('Минимальный индекс структурного различия в обучающем множестве',min\_iss\_train)

ttl = 'Пятиугольник'

cls = 2

plotData(train\_data, ttl, cls)

**Арксинус**

def parab(x, a,b):

x\_noise = np.random.uniform(-0.00001, 0.00001)

y\_noise = np.random.uniform(-0.1, 0.1)

y = a\*np.arcsin(b\*(x +x\_noise )) + y\_noise

return y

#

def one\_class(n, fn, fn2):

file = open(fn, 'wb')

file2 = open(fn2, 'wb')

xe = w2 - border - 1

dy = 0.01

ys = 0

label = 0

for i in range(n): # n - число примеров

sgn = np.random.randint(2) # y или -y

a = np.random.randint(-10, 10)

b = np.random.uniform(0.1, 1)

arrPic = np.zeros((w, h), dtype = np.uint8)

y = ys - dy

while (y+dy) < 1/b:

y += dy

x = parab(y, a, b)

iy = min(w - 1, int(w2 + y))

ix = min(w - 1,int(w2 - x))

clr = np.random.randint(clr\_mim, clr\_max)

arrPic[ix, iy] = clr

arrPic[-ix, -iy] = clr

file.write(arrPic)

file2.write(np.uint8(label))

file.close()

file2.close()

**Арксинус 1/2**

def parab(x, a,b):

x\_noise = np.random.uniform(-0.5, 0.5)

y\_noise = np.random.uniform(-10, 10)

y = a\*np.arcsin(b\*(x)) + border + y\_noise

return y

#

def one\_class(n, fn, fn2):

file = open(fn, 'wb')

file2 = open(fn2, 'wb')

xe = w2 - border - 1

dy = 0.01

ys = 0

label = 0

for i in range(n): # n - число примеров

sgn = np.random.randint(2) # y или -y

a = np.random.randint(1, 10)

b = np.random.uniform(0.1, 1)

arrPic = np.zeros((w, h), dtype = np.uint8)

y = ys - dy

while (y+dy) < 1/b:

y += dy

x = parab(y, a, b)

iy = min(w - 1, int(w2 + y))

ix = min(w - 1,int(w2 - x))

clr = np.random.randint(clr\_mim, clr\_max)

if sgn == 1:

arrPic[ix, iy] = clr

else:

arrPic[-ix, -iy] = clr

file.write(arrPic)

file2.write(np.uint8(label))

file.close()

file2.close()

**Декартов лист**

def figure(a,t):

t\_noise = np.random.uniform(-4.1,4.1)

#t\_noise=0

x = a \* ((t+t\_noise)\*\*2 - 1) / (3 \* (t+t\_noise)\*\*2 + 1)

y = a \* (t+t\_noise) \* ((t+t\_noise)\*\*2 - 1) / (3 \* (t+t\_noise)\*\*2 + 1)

for i in range(len(x)):

x\_noise = np.random.uniform(-13,13)

x[i] += x\_noise

y\_noise = np.random.uniform(-0.5,0.5)

y[i] += y\_noise

return x,y

#

def one\_class(n, fn, fn2):

file = open(fn, 'wb')

file2 = open(fn2, 'wb')

label = 0

t = np.linspace(-5, 5, 400)

for i in range(n): # n - число примеров

a = np.random.randint(-20,21)

x,y = figure(a,t)

x\_min = int(min(x))

x\_max = int(max(x))

y\_min = int(min(y))

y\_max = int(max(y))

dx = int((64 - (x\_max - x\_min)) / 2) # Половина свободного пространства по x

dy = int((64 - (y\_max - y\_min)) / 2) # Половина свободного пространства по y

shift\_x = dx - x\_min # Сдвиг по x

shift\_y = dy - y\_min # Сдвиг по y

w = h = 64 # Ширина и высота рисунка

arrPic = np.zeros((w, h), dtype = np.uint8)

clr\_mim, clr\_max = 75, 255 # Диапазон оттенков серого цвета

for x, y in zip(x, y):

ix = int(x) + shift\_x

iy = int(y) + shift\_y

clr = np.random.randint(clr\_mim, clr\_max)

arrPic[iy, ix] = clr

file.write(arrPic)

file2.write(np.uint8(label))

file.close()

file2.close()

**Локон Аньези**

def lokon\_anezi(a,t):

x\_noise = np.random.uniform(-0.5, 0.5)

y\_noise = np.random.uniform(-10, 10)

x = 2 \* a / np.tan(t) + x\_noise

y = 2 \* a \* np.sin(t)\*\*2 + border + y\_noise

return x,y

#

def one\_class(n, fn, fn2):

file = open(fn, 'wb')

file2 = open(fn2, 'wb')

label = 0

t = np.linspace(0.1, np.pi - 0.1, 100)

for i in range(n): # n - число примеров

sgn = np.random.randint(2) # x или -x

a = np.random.randint(1,20)

x,y = lokon\_anezi(a,t)

arrPic = np.zeros((w, h), dtype = np.uint8)

j=0

while (x[j]>0):

ix1 = min(w - 1, int(w2 + x[j]))

ix2 = max(0, int(w2 - x[j]))

iy = h - int(y[j])

iy = max(0, iy)

iy = min(h - 1, iy) # Уходим из физической системы координат

clr = np.random.randint(clr\_mim, clr\_max)

arrPic[iy, ix1] = clr

arrPic[iy, ix2] = clr

j+=1;

file.write(arrPic)

file2.write(np.uint8(label))

file.close()

file2.close()

**Программа визуализации**

def plotData(data, ttl, cls):

plt.figure(ttl)

k = 0

for i in range(10):

j = np.random.randint(data.shape[0])

k += 1

plt.subplot(1, 10, k)

plt.imshow(data[i], cmap = 'gray')

plt.title(cls, fontsize = 11)

plt.axis('off')

plt.subplots\_adjust(hspace = -0.1) # wspace

plt.show()

ttl = 'Декартов Лист' # например

cls = 2

plotData(train\_data, ttl, cls)

**Проверка на неповторяемость**

def iss(data):

m = 0.5

s\_iss=[]

cnt = 0

for i in range(len(data)):

im1 = data[i].reshape(w\*h)

for j in range(i + 1, len(data)):

im2 = data[j].reshape(w\*h)

iss = (1-abs(structural\_similarity(im1, im2)))/2

if iss < m:

m = iss

s\_iss.append(m)

return sum(s\_iss[len(s\_iss)-int(len(s\_iss)/3):])/int(len(s\_iss)/3)

def check\_sim(data,iss):

n = len(data)

cnt = 0

for i in range(n):

im1 = data[i].reshape(w\*h)

for j in range(i + 1, n):

im2 = data[j].reshape(w\*h)

d = (1-structural\_similarity(im1, im2))/2

if d < iss:

cnt+=1

return cnt##################################################################################

min\_iss\_test=iss(test\_data)

print('Минимальный индекс структурного различия в проверочном множестве',min\_iss\_test)

print('Проверка на неповторяемость', check\_sim(test\_data, min\_iss\_test ))

min\_iss\_train=iss(train\_data)

print('Минимальный индекс структурного различия в обучающем множестве',min\_iss\_train)

print('Проверка на неповторяемость', check\_sim(train\_data, min\_iss\_train))