Валентина Старикова

А-05-18 | Вариант 21

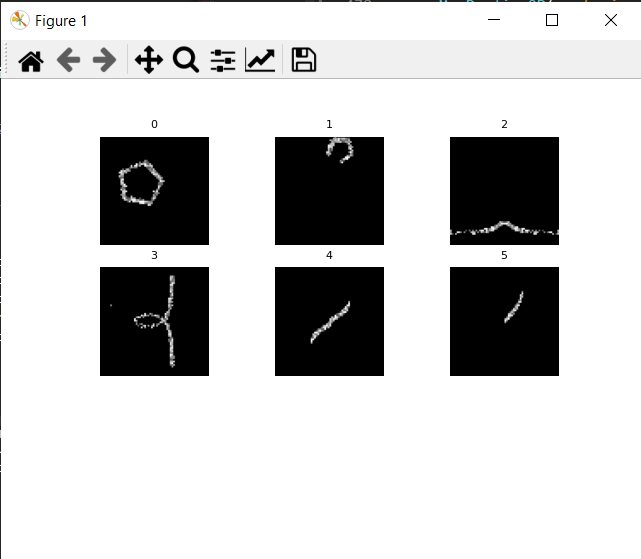
Отчет по лабораторной работе 8

часть 2

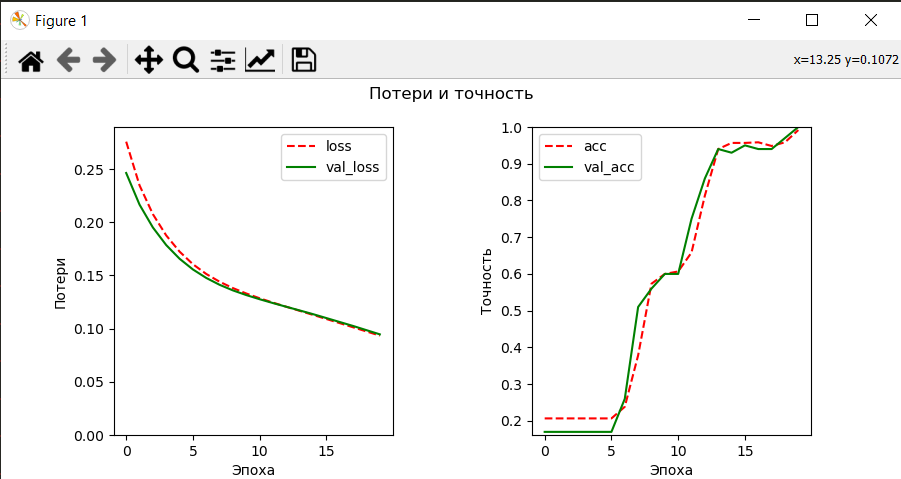
1. Условие решаемой задачи.

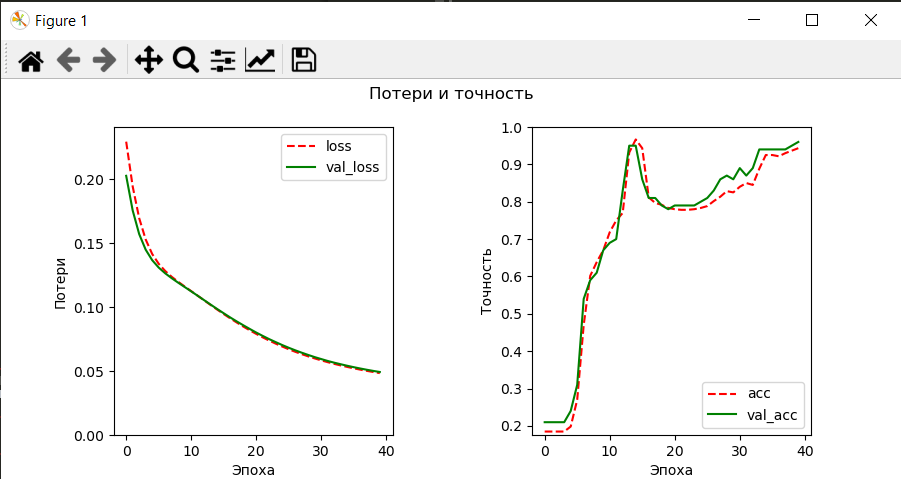
Реализация нейронной сети, позволяющей классифицировать изображения, из созданного набора данных. Ищется минимальная архитектура сети, обеспечивающей 100% точности классификации (или близкую к ней).

1. Примеры загруженных изображений (по одному из каждого класса).



1. Описание (символьное) модели НС, обеспечивающей точность классификации 100 % на оценочном множестве, графики обучения этой НС.





1. Описание (символьное) модели НС с минимальной архитектурой, обеспечивающей ту же точность классификации, что и прежняя НС (см. п. выше), и графики обучения второй сети.

|  |  |  |  |
| --- | --- | --- | --- |
| **Вариант** | **Время вычислений** | **Точность на обучающей выборке** | **Точность на оценочной выборке** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. Программы загрузки набора данных, визуализации изображений, создания и обучения НС.

**Программы загрузки набора данных**

#Пятиугольник

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

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

y\_noise = np.random.uniform(-1,1)

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

arrPic[i, y] = np.random.randint(clr\_mim, clr\_max)

def pent(arrPic, full=True):

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 Pentagon(full=True):

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

pent(arrPic,full)

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

arrPic = rot\_img(ang, arrPic)

return arrPic

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

#Локон Аньези

def lokon\_anezi(a,t):

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

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

for i in range(len(x)):

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

x[i] += x\_noise

y\_noise = np.random.uniform(-2,2)

y[i] += y\_noise + 8

return x,y

def Witch\_of\_Agnesi():

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

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

return arrPic

#Декартов лист

def figure(a,t):

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

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

for i in range(len(x)):

x\_noise = np.random.uniform(-1,2)

x[i] += x\_noise

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

y[i] += y\_noise

return x,y

def Folium\_of\_Descartes():

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

a = np.random.randint(4,21)

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)

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

arrPic[-ix, -iy] = clr

return arrPic

#Арксинус

def arcs(x, a,b):

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

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

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

return y

def Half\_of\_Arcsine():

xe = w2 - border - 1

dy = 0.01

ys = 0

label = 0

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 = arcs(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

return arrPic

def Arcsine():

xe = w2 - border - 1

dy = 0.01

ys = 0

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 = arcs(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

arrPic[-ix, -iy] = clr

return arrPic

#--------------------------------------------------Создание набора данных-----------------------------------------------------

def clss(c):

if c==0:

arrpic = Pentagon()

elif c==1:

arrpic = Pentagon(False)

elif c==2:

arrpic = Witch\_of\_Agnesi()

elif c==3:

arrpic = Folium\_of\_Descartes()

elif c==4:

arrpic = Arcsine()

elif c==5:

arrpic = Half\_of\_Arcsine()

return arrpic

def create\_dataset(n, fn, fnl, clss\_inst):

file = open(fn, 'wb')

file2 = open(fnl, 'wb')

for i in range(n):

c = np.random.randint(0, 6)

clss\_inst[c]+=1

arrpic = clss(c)

file.write(arrpic)

file2.write(np.uint8(c))

file.close()

file2.close()

class\_instance = np.zeros(6)

create\_dataset(n\_test, fn\_test, fn\_test\_labels, class\_instance)

print('Количество экземпляров в классах в тестовом множестве',class\_instance)

create\_dataset(n\_train, fn\_train, fn\_train\_labels, class\_instance)

print('Количество экземпляров в классах в обучающем множестве',class\_instance)

**Bизуализации изображений**

#--------------------------Примеры загруженных изображений (по одному из каждого класса)-----------------------------------

if show:

for i in range(6):

arrpic = clss(i)

plt.subplot(3, 3, i+1)

plt.imshow(arrpic, cmap = 'gray')

plt.title(i, fontsize = 8)

plt.axis('off')

plt.subplots\_adjust(wspace = 0.2) # wspace

plt.show()

**Cоздание и обучение НС**

# Обучение нейронной сети

epochs = 20

start = time.time()

history = model.fit(x\_train, y\_train, batch\_size = 128, epochs = epochs,

verbose = 2, validation\_data = (x\_test, y\_test), callbacks = callbacks)