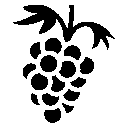
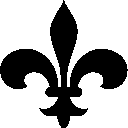
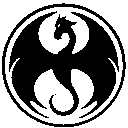
**Лущик Никита КС-44**

**Нейронная сеть Коско**

**Входные данные**

На вход программы в качестве эталонных образов необходимо подать несколько изображений произвольного цвета и размера – для ускорения обучения, в нашем случае это будут следующие 3 монохромных изображения, по 128х128 пикселей каждое:

0)  1)  2) 

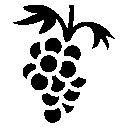
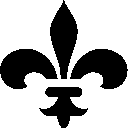
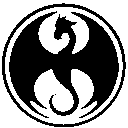
Эти изображения, в свою очередь, становятся массивами значений в диапазоне [-1; 1], где -1 – белый пиксель, 1 – черный, на которых и обучается нейронная сеть. Соотношение между количеством бинарных признаков, характеризующих образы, и количеством самих запоминаемых образов, выполняется:

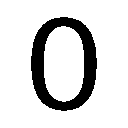
**3 ≤ 2458**

В качестве выходных примеров используем, соответственно, следующие изображения 64х64:

0)  1)  2) 

А в качестве тестовой выборки подадим уже обработанные и использовавшиеся при обучении образы, но зашумленные, и 1 новый образ:

0)  1)  2)  3)



**Результат**

После обучения сеть успешно опознала зашумленные образы 0, 1 и 2, и не смогла распознать образ 3, т.к. образа не было в обучающей выборке. Соответствующие выходные образы могут быть преобразованы обратно в изображения и использованы, если это необходимо.



Вывод пример 0



Вывод пример 1



Вывод пример 2



Вывод пример 3

**Листинг AutoassociativeMemory.py**

import Common

from PIL import Image

from numpy import array

def imageRecognizer(filename):

img = Image.open(filename)

img = img.convert('1')

return list(array(img))

def binArrayParser(array):

returnArray = list()

for i in array:

for j in i:

if (j == False):

returnArray.append(1)

elif (j == True):

returnArray.append(-1)

return returnArray

def Cosco(X, Y, X0):

W = list(Common.multiply(Common.transpose(X), Y))

#Common.NumerizedPrint(W)

f = open("log.txt", "w")

for e in range(len(X0)):

S1 = list(Common.multiply(Common.transpose(W), Common.transpose([X0[e]])))

#Common.NumerizedPrint(S1)

S2 = list()

Y1 = list()

X1 = list()

counter = 0

while True:

print("Iteration ", counter)

f.write(f"\nIteration {counter}")

Yprev = Y1.copy()

Y1.clear()

for i in range(len(S1)):

for j in range(len(S1[0])):

Y1.append([Common.SingleJump(S1[i][j], 0)])

#print(Y1)

f.write(f"\nY1:\n {Y1}")

S2 = Common.multiply(W, Y1)

#print(S2)

f.write(f"\nS2:\n {S2}")

for i in range(len(S2)):

for j in range(len(S2[0])):

X1.append([Common.SingleJump(S2[i][j], 0)])

#print(X1)

f.write(f"\nX1:\n {X1}")

if (counter > 0):

n = sum(Common.powList(Common.difference2dim(Y1, Yprev)))

if (n == 0):

switcher = False

print("n = 0, this means that outputs are stabilized")

f.write("\nn = 0, this means that outputs are stabilized")

for i in range(len(Y)): # 2.2

if (Y[i] == Common.dim2to1(Y1)):

f.write(f"\nMatch to training image{i}")

print("Match to training image", i)

switcher = True

if (switcher == False):

print("No Matches to training images")

f.write(f"\nNo Matches to training images")

break

S1 = Common.multiply(Common.transpose(W), X1)

#Common.NumerizedPrint(S1)

counter += 1

X1.clear()

f.close()

def main():

#inputArray = [[1, -1, -1, 1, -1, 1, -1, -1, -1, -1, 1, -1, 1, -1, -1, 1],

# [1, 1, 1, 1, 1, -1, -1, 1, 1, -1, -1, 1, 1, 1, 1, 1]]

#outputArray = [[1, 1, 1, 1, -1, 1, 1, -1, 1],

# [1, -1, 1, 1, 1, -1, 1, -1, 1]]

#testArray = [[1, 1, -1, 1, -1, 1, -1, -1, 1, -1, 1, 1, -1, -1, -1, 1],

# [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]]

inputArray = list()

outputArray = list()

testArray = list()

inputArray.append(binArrayParser(imageRecognizer('mono128x128/img3mono.bmp')))

inputArray.append(binArrayParser(imageRecognizer('mono128x128/img4mono.bmp')))

inputArray.append(binArrayParser(imageRecognizer('mono128x128/img5mono.bmp')))

outputArray.append(binArrayParser(imageRecognizer('mono128x128/imgOut1.bmp')))

outputArray.append(binArrayParser(imageRecognizer('mono128x128/imgOut2.bmp')))

outputArray.append(binArrayParser(imageRecognizer('mono128x128/imgOut3.bmp')))

testArray.append(binArrayParser(imageRecognizer('mono128x128/img3mono.bmp')))

testArray.append(binArrayParser(imageRecognizer('mono128x128/img4mononoisy.bmp')))

testArray.append(binArrayParser(imageRecognizer('mono128x128/img5mononoisy.bmp')))

testArray.append(binArrayParser(imageRecognizer('mono128x128/img6mononoisy.bmp')))

Cosco(inputArray, outputArray, testArray)

main()

**Листинг Common.py**

def SingleJump(S, T):

if (S <= T):

return -1

elif (S > T):

return 1

def transpose(array):

transposed = list()

for i in range(len(array[0])):

row = list()

for sublist in array:

row.append(sublist[i])

transposed.append(row)

return transposed

def multiply(array0, array1):

multiplied = []

counter = len(array0)

for k in range(len(array0)):

stringsum = []

for i in range(len(array1[0])):

string = 0

for j in range(len(array0[0])):

string += array0[k][j]\*array1[j][i]

stringsum.append(string)

if ((k % 100) == 0):

print(k, " from", counter)

multiplied.append(stringsum.copy())

return multiplied

def difference2dim(array0, array1):

diff = list()

for i in range(len(array0)):

for j in range(len(array0[0])):

diff.append(array0[i][j] - array1[i][j])

return diff

def dim2to1(array0):

dim = list()

for i in range(len(array0)):

for j in range(len(array0[0])):

dim.append(array0[i][j])

return dim