**Лущик Никита КС-44**

**Нейронная сеть Хемминга**

**Входные данные**

На вход программы в качестве эталонных образов подадим несколько изображений – в нашем случае это будут следующие 3 монохромных изображения, по 4х4 пикселей каждое:

0)  1)  2) 

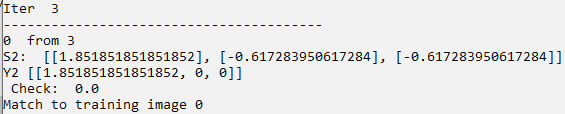
Эти изображения, в свою очередь, становятся массивами значений в диапазоне [-1; 1], где -1 – белый пиксель, 1 – черный, на которых и обучается нейронная сеть.

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

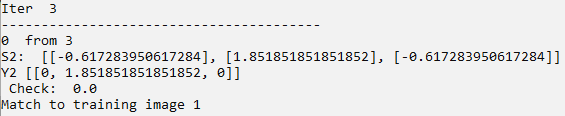
0)  1)  2) 

**Результат**

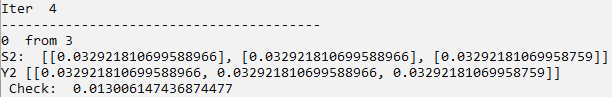
После обучения сеть успешно опознала зашумленные образы 0, 1 и не смогла распознать образ 3 – похоже, для сети принципиально наличие всех пикселей, составляющих обучающий образ.



Вывод пример 0



Вывод пример 1

****

Вывод пример 2

**Листинг Hamming.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 Hamming(X, Xn):

print(X)

W = list(Common.multiplier(X, 0.5))

print(W)

T = len(W[0]) / 2

eps = 1 / len(X)

E = [[-eps for j in range(len(X))] for i in range(len(X))]

E = Common.diagonalNullifier(E, 1)

print(E)

Emax = 0.1

for e in range(len(Xn)):

print("\nImage", e)

S1 = Common.summator(Common.multiply(W, Common.transpose([Xn[e]])), T)

Y1 = list([[Common.LinearThreshold(S1[i][0], T) for i in range(len(S1))]])

print("S1:", S1, "\n\nY1:", Y1, "\n2")

Y2 = Y1

Yprev = Y1

check = 1

counter = 0

while(check >= Emax):

print("\nIter ", counter, "\n----------------------------------------")

Yprev = Y2

S2 = Common.multiply(E, Common.transpose(Y2))

Y2 = list([[Common.LinearThreshold(S2[i][0], T) for i in range(len(S2))]])

check = sum(Common.powList(Common.difference2dim(Y2, Yprev)))

counter += 1

print("S2: ", S2, "\nY2", Y2, "\n", "Check: ", check)

print("Match to training image", Y2[0].index(max(Y2[0])))

def main():

inputArray = list()

testArray = list()

inputArray.append(binArrayParser(imageRecognizer('mono128x128/img3.bmp')))

inputArray.append(binArrayParser(imageRecognizer('mono128x128/img4.bmp')))

inputArray.append(binArrayParser(imageRecognizer('mono128x128/img5.bmp')))

testArray.append(binArrayParser(imageRecognizer('mono128x128/img3noisy.bmp')))

testArray.append(binArrayParser(imageRecognizer('mono128x128/img4noisy.bmp')))

testArray.append(binArrayParser(imageRecognizer('mono128x128/img5noisy.bmp')))

Hamming(inputArray, testArray)

main()

**Листинг Common.py**

def LinearThreshold(S, T):

if (S <= 0):

return 0

elif (S > 0 and S <= T):

return S

elif (S >= T):

return T

def multiply(array0, array1, adding = 0):

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] + adding

stringsum.append(string)

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

print(k, " from", counter)

multiplied.append(stringsum.copy())

return multiplied

def summator(array, summ):

result = array

for i in range(len(result)):

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

result[i][j] = result[i][j] + summ

return result

def multiplier(array, mult):

multi = array

for i in range(len(multi)):

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

multi[i][j] = multi[i][j]\*mult

return multi