Гайворонский Сергей Игоревич, группа 2.1

Лабораторная работа № 1

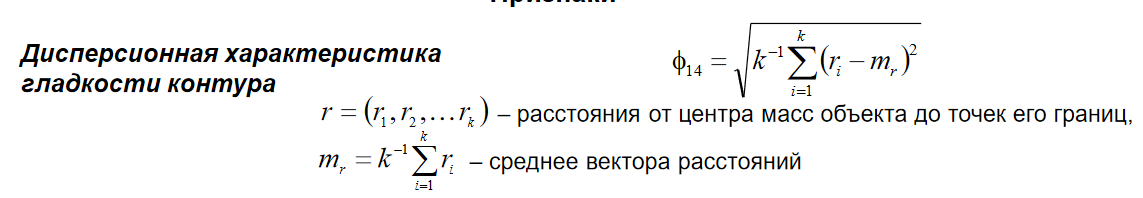
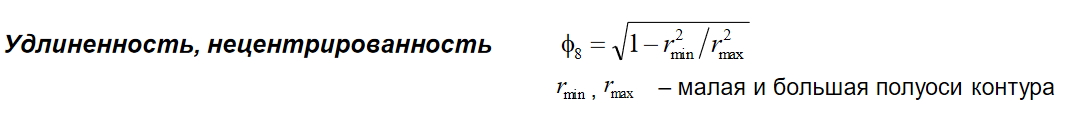
**Классификация объектов с использованием mlp-сетей**

**Задание**

Реализовать нейросетевой классификатор исходных изображений круп, представленных в виде смеси на фотографии, используя библиотеку OpenCV или ее аналоги.

**Результат выполнения задания**

Использованные признаки классификации:



Кодирование признаков:

def borderContrast(imgBin, imgGray, contour):

    kernel = np.ones((5, 5), np.uint8)

    imgBrdOut = cv2.morphologyEx(imgBin, cv2.MORPH\_DILATE, kernel, iterations=1)

    imgBrdOut = cv2.morphologyEx(imgBrdOut, cv2.MORPH\_GRADIENT, kernel, iterations=1)

    cv2.imwrite('sample\_data/imgBrdOut.png', imgBrdOut)

    imgBrdIn = cv2.morphologyEx(imgBin, cv2.MORPH\_ERODE, kernel, iterations=1)

    imgBrdIn = cv2.morphologyEx(imgBrdIn, cv2.MORPH\_GRADIENT, kernel, iterations=1)

    cv2.imwrite('sample\_data/imgBrdIn.png', imgBrdIn)

    meanBrdOut, stddevBrdOut = cv2.meanStdDev(imgGray, mask=imgBrdOut)

    meanBrdIn, stddevBrdIn = cv2.meanStdDev(imgGray, mask=imgBrdIn)

    if meanBrdIn[0][0] > meanBrdOut[0][0]:

        return abs(meanBrdIn[0][0] - meanBrdOut[0][0]) / (meanBrdIn[0][0]+1)

    else:

        return abs(meanBrdIn[0][0] - meanBrdOut[0][0]) / (meanBrdOut[0][0]+1)

def elongation(contour):

    m = cv2.moments(contour)

    cx = int(m['m10']/m['m00'])

    cy = int(m['m01']/m['m00'])

    max = 0

    min = 10000000000000000000

    for kp in contour:

        r = math.sqrt((kp[0][0] - cx) \*\* 2 + (kp[0][1] - cy) \*\* 2)

        if (r >= max):

            max = r

        if (r <= min):

            min = r

    fi = math.sqrt(1 - min \*\* 2 / max \*\* 2)

    return fi

def contoursmoothness(contour):

    kpCnt = len(contour)

    x = 0

    y = 0

    for kp in contour:

        x = x+kp[0][0]

        y = y+kp[0][1]

    x = x / kpCnt

    y = y / kpCnt

    sum\_r = 0

    for kp in contour:

        sum\_r += math.sqrt((kp[0][0] - x) \*\* 2 + (kp[0][1] - y) \*\* 2)

    m = sum\_r / kpCnt

    sum\_rm = 0

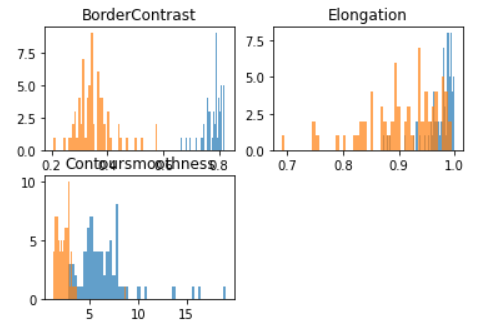
    for kp in contour:

        sum\_rm += (math.sqrt((kp[0][0] - x) \*\* 2 + (kp[0][1] - y) \*\* 2) - m) \*\* 2

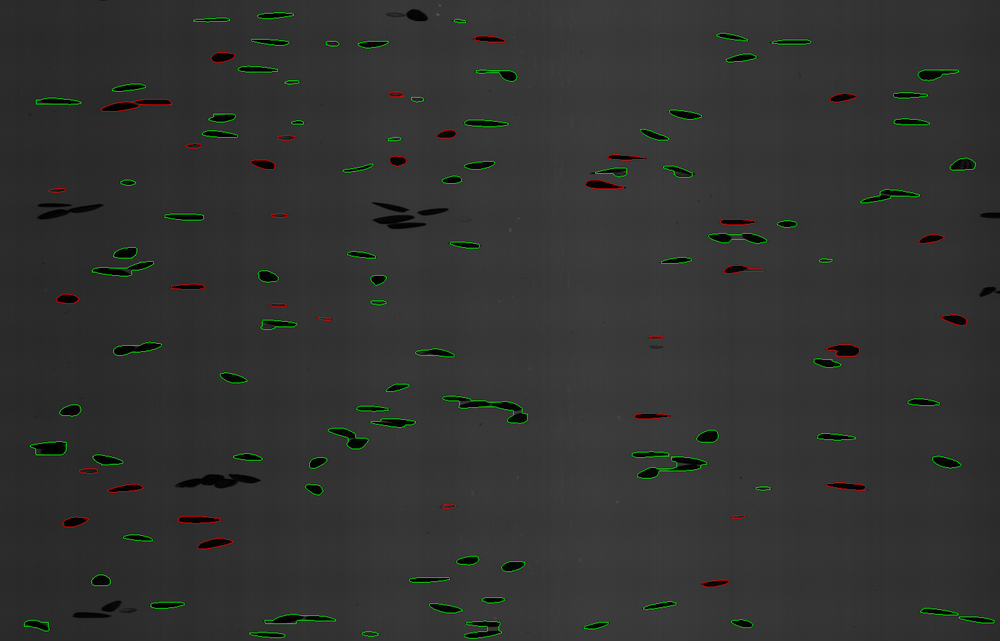
    fi = math.sqrt(sum\_rm / kpCnt)

    return fi

Гистограммы признаков:



Итоговая классификация



Точность и функция потерь при обучении нейронной сети:



# Вывод

В ходе выполнения работы были выполнены все задания.