Задание

1. Выполнить максимально возможный диагональный сдвиг вниз буквы набора данных EMNIST.

Привести исходное и результирующее изображения.

Использовать тестовое множество EMNIST.

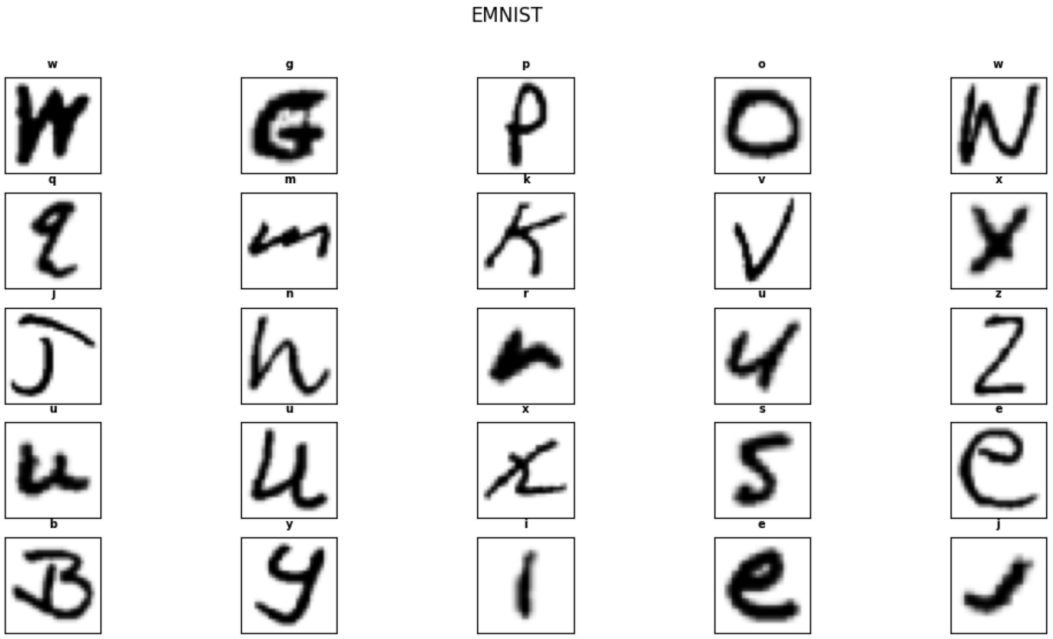
Алгоритм

1. Посчитать максимальное количество пикселей, на которые можно сдвинуть вниз.
2. Посчитать максимальное количество пикселей, на которые можно сдвинуть влево.
3. Применить горизонтальный сдвиг влево
4. Применить вертикальный сдвиг вниз

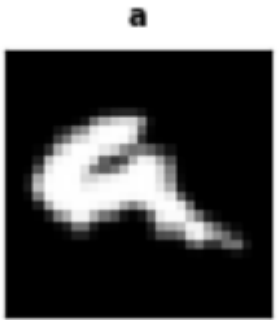
Входные данные

Набор данных содержит следующие изображения:

- EMNIST-letters – 145'600 рукописных букв английского алфавита; из них 124'800 входят в обучающую выборку, а 20'800 – в тестовую; размер каждого образа – 28*28 пикселей; рисунки выполнены в оттенках серого цвета;



Выбранная буква



Выходные данные

```
number of classes:  26
emnist_letters-данные сохранены в двоичные файлы
Загрузка данных из двоичных файлов
на столько пикселей можно сдвинуть слева 6
на столько пикселей можно сдвинуть вниз 2
```

Программа

```
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
import numpy as np
import matplotlib.pyplot as plt
from tensorflow import keras
from mnist import MNIST
from scipy.spatial import distance
from skimage.metrics import structural_similarity as compare_ssim # pip
install scikit-image
n= 'emnist_letters'
num_classes = 26
x_train_buf_number = 124800 #обучающая выборка
x_test_buf_number = 20800 #выборка валидации
map_for_emnist = dict(zip(np.arange(0, 26),['a', 'b', 'c', 'd', 'e',
'f', 'g', 'h',
                                            'i', 'j', 'k', 'l', 'm', 'n',
'o', 'p',
                                            'q', 'r', 's', 't', 'u', 'v',
'w', 'x',
```

```
                                          'y', 'z']))
img_rows = img_cols = 28
#dist = distance.euclidean(im1, im2) # Евклидово расстояние
#dist_cs = distance.cosine(im1, im2) # Косинусное расстояние
#sim = compare_ssim(im1, im2) # Индекс структурного сходства изображений

#запись данных в бинарные файлы
def DatatoBIN(n):
    buf_path='C:\\Users\\Мария\\PycharmProjects\\resources\\'
    #buf_path= "C:\\Users\\Мария\\PycharmProjects\\Lab6\\lab6"
    buf_path = buf_path + '\\' + n
    mndata = MNIST(buf_path)
    mndata.gz = True
    imagesTrain, labelsTrain = mndata.load_training()
    imagesTest, labelsTest = mndata.load_testing()
    f1 = open(buf_path +'imagesTrain.bin', 'wb')
    f2 = open(buf_path +'labelsTrain.bin', 'wb')
    f3 = open(buf_path +'imagesTest.bin', 'wb')
    f4 = open(buf_path + 'labelsTest.bin', 'wb')
    f1.write(np.uint8(imagesTrain))
    f2.write(np.uint8(labelsTrain))
    f3.write(np.uint8(imagesTest))
    f4.write(np.uint8(labelsTest))
    f1.close()
    f2.close()
    f3.close()
    f4.close()
    print(n+'-данные сохранены в двоичные файлы')
#запись данных из двоичных файлов в массивы
def ReadBIN(n):
    buf_path='C:\\Users\\Мария\\PycharmProjects\\resources\\'
    buf_path = buf_path + '\\' + n
    print('Загрузка данных из двоичных файлов')
    with open(buf_path + 'imagesTrain.bin', 'rb') as read_binary:
        img1 = np.fromfile(read_binary, dtype = np.uint8)
    with open(buf_path + 'labelsTrain.bin', 'rb') as read_binary:
        label1 = np.fromfile(read_binary, dtype = np.uint8)
    with open(buf_path + 'imagesTest.bin', 'rb') as read_binary:
        img2 = np.fromfile(read_binary, dtype = np.uint8)
    with open(buf_path + 'labelsTest.bin', 'rb') as read_binary:
        label2 = np.fromfile(read_binary, dtype = np.uint8)
    return img1, label1, img2, label2
def OutputData25(x_train):
    x_buf = x_train.copy()
    y_buf = y_train.copy()
    name = "EMNIST"
```

```
    plt.figure(figsize=(10, 10))
    plt.suptitle(name)
    for i in range(25):
        plt.subplot(5, 5, i + 1)
        plt.xticks([])
        plt.yticks([])
        plt.imshow(x_buf.reshape(-1, 28, 28, 1).transpose(0, 2, 1, 3)[i],
cmap=plt.cm.binary)
        plt.title(map_for_emnist[y_buf[i]], size=7, weight="heavy")
    plt.show()

def OutputData1(x_train):
    x_buf = x_train.copy()
    y_buf = y_train.copy()
    name = "EMNIST"
    plt.figure(figsize=(10, 10))
    plt.suptitle(name)
    for i in range(1):
        plt.subplot(5, 5, i + 1)
        plt.xticks([])
        plt.yticks([])
        plt.imshow(x_buf.reshape(-1, 28, 28, 1).transpose(0, 2, 1, 3)[i],
cmap=plt.cm.binary)
        plt.title(map_for_emnist[y_buf[i]], size=7, weight="heavy")
    plt.show()

def OutputData16(x, y):
    x_buf = x.copy()
    y_buf = y.copy()
    name = "EMNIST"
    plt.figure(figsize=(10, 10))
    plt.suptitle(name)
    for i in range(25):
        plt.subplot(5, 5, i + 1)
        plt.xticks([])
        plt.yticks([])
        plt.imshow(x_buf.reshape(-1, 28, 28, 1).transpose(0, 2, 1, 3)[i],
cmap='gray')
        plt.title(map_for_emnist[y_buf[i]], size=7, weight="heavy")
    plt.show()

def buf_x_y(num_classes):
    buf = 1
    x_train = np.asarray(imagesTrain)
    y_train = np.asarray(labelsTrain)-buf
    x_test = np.asarray(imagesTest)
```

```
    y_test = np.asarray(labelsTest)-buf
    x_train = np.array(x_train, dtype='float32') / 255 #стандартизация
    x_test = np.array(x_test, dtype='float32') / 255
    # преобразование в вектор, размерность которого равна кол-ву классов
задачи
    y_train_cat = keras.utils.to_categorical(y_train, num_classes)
    y_test_cat = keras.utils.to_categorical(y_test, num_classes)
    return x_train, y_train, x_test, y_test, y_train_cat, y_test_cat

#-------------------------------------------------main---------------
------------------------------
print("_____________________________________________________________")
print("number of classes: ", num_classes)
DatatoBIN(n)
imagesTrain, labelsTrain, imagesTest, labelsTest = ReadBIN(n)
x_train, y_train, x_test, y_test, y_train_cat, y_test_cat =
buf_x_y(num_classes)
#OutputData25(x_train)
#1. Выполнить максимально возможный диагональный сдвиг вниз буквы набора
данных EMNIST.
#Привести исходное и результирующее изображения.
#Использовать тестовое множество EMNIST.
OutputData16(x_test, y_test)

x_train = x_train.reshape(-1, img_rows, img_cols)
x_test = x_test.reshape(-1, img_rows, img_cols)


def shiftdown(x): #на сколько можно сдвинуть снизу
    #print(x.shape)
    fl = True
    i = 27
    while (fl) and (i>=0):
        buf = np.sum(x[i])
        #print("xi", x[i])
        if (buf > 0):
            fl = False
        else:
            i-=1
    return (27-i)

def shiftleft(x1): #на сколько можно сдвинуть слева
    #print(x1.shape)
    i = 0
    j=0
    for j in range(28):
```

```
        buf = 0
        for i in range (28):
            buf+= x1[i][j]
            #print(buf)
        if (buf>0):
            #print (j)
            return j



x_test = x_test.reshape(-1, img_rows, img_cols)
i = 0
#print(x_test.shape)
h_left = shiftleft(x_test[i])
print("на столько пикселей можно сдвинуть слева", h_left)


x_test = x_test.reshape(-1, img_rows, img_cols)
i = 0
h_down=shiftdown(x_test[i])
print("на столько пикселей можно сдвинуть вниз", h_down)


#x_new = shift(x_test[i], h_left, h_down)
#print(x_new)
```