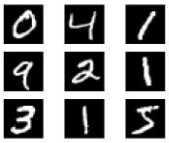
Тестовые задания на вакансию Разработчик в R&D, Azoft

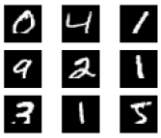
1. Распознавание частично закрытых рукописных цифр MNIST

Дано: Есть рукописные цифры MNIST

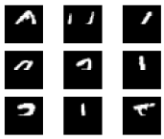


Каждая картинка имеет размер 28x28. Задача – распознать картинки с помощью сверточной нейронной сети. При этом нужно рассмотреть также случаи с частично закрытыми картинками:

Цифры открыты на 75% (пиксели с y = 21…27 занулены)



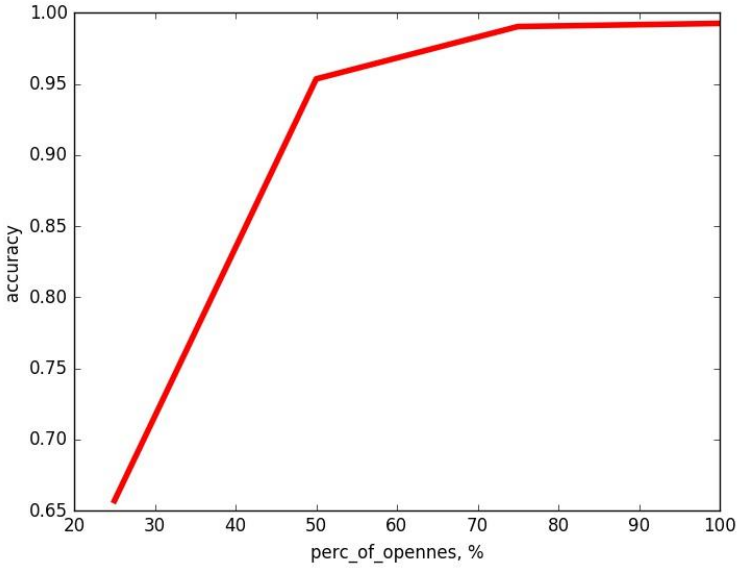
Цифры открыты на 50% (пиксели с y = 14…27 занулены)



Нужно построить график точности (accuracy) в зависимости от степени открытости цифр. Должен быть посчитан хотя бы для 4х точек: 25% 50% 75% 100%

Решение:

Процент открытости цифр	**accuracy**
100%	0.99250000000000005
75%	0.99029999999999996
50%	0.95350000000000001
25%	0.65710000000000002



Текст программы:

```
import numpy as np
import matplotlib.pyplot as plt
from keras.models import Sequential
#слои
from keras.layers import Dense, Dropout, Flatten
#сверточные слои
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.utils import np_utils
#данные
from keras.datasets import mnist

from keras import backend as K

'''
Частичное сокрытие цифр
dataset - набор данных, который модифицируется
shape_dataset - размерность
пиксели с y = start_ind…end_ind занулены
'''
def hiding_digits(dataset, shape_dataset, start_ind, end_ind):
    for i_train in range(shape_dataset):
        for i in range(start_ind, end_ind):
            for j in range(28):
                if dataset[i_train, 0, i, j] != 0.:
                    dataset[i_train, 0, i, j] = 0.
    return dataset


'''
создание и компиляция модели
'''
def create_model():
   model = Sequential()
    #слой свертки, 32 карты признаков, размер ядра свертки = (5,5)
   model.add(Conv2D(32, kernel_size=(5, 5), input_shape=(1, 28, 28),
activation='relu'))
   #слой подвыборки, размер пула = (2, 2)
   model.add(MaxPooling2D(pool_size=(2, 2)))
    # слой свертки, 32 карты признаков, размер ядра свертки = (5,5)
   model.add(Conv2D(32, kernel_size=(2, 2), activation='relu'))
    # слой подвыборки, размер пула = (2, 2)
   model.add(MaxPooling2D(pool_size=(2, 2)))
    # слой исключения
   model.add(Dropout(0.2))
   model.add(Flatten())
    #полносвязный слой, 400 нейронов
   model.add(Dense(400, activation='relu'))
    #полносвязный выходной слой, 10 нейронов, которые соответствуют классам рукописных
цифр от 0 до 9.
   model.add(Dense(10, activation='softmax'))
    # Компилируем модель
   model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
   return model


# Цифры открыты на 75% (пиксели с y = 21…27 занулены)
# Цифры открыты на 50% (пиксели с y = 14…27 занулены)
# Цифры открыты на 25% (пиксели с y = 7…27 занулены)
def create_CNN(flag, X_dataset_train, X_dataset_test, y_train, y_test, start_ind=None,
```

```
end_ind=None):
    if flag == 0:
        X_train_new = X_dataset_train
        X_test_new = X_dataset_test
    else:
        X_train_new = hiding_digits(X_dataset_train, X_dataset_train.shape[0],
start_ind, end_ind)
        X_test_new = hiding_digits(X_dataset_test, X_dataset_test.shape[0], start_ind,
end_ind)

    # строим модель
    model = create_model()
    # обучаем модель
    model.fit(X_train_new, y_train, validation_data=(X_test_new, y_test), epochs=20,
batch_size=32, verbose=2,
                        validation_split=0.2)
    # качество обучения сети на тестовой выборке
    scores = model.evaluate(X_test_new, y_test, verbose=1)[1]
    return scores

if __name__ == '__main__':
    K.set_image_dim_ordering('th')
    # загружаем набор данных
    (X_train, y_train), (X_test, y_test) = mnist.load_data()

    # меняем размерность (N, глубина - черно белая картинка, высота, ширина)
    X_train = X_train.reshape(X_train.shape[0], 1, 28, 28).astype('float32')
    X_test = X_test.reshape(X_test.shape[0], 1, 28, 28).astype('float32')

    # нормализация данных
    X_train /= 255
    X_test /= 255
    # Преобразуем метки в категории
    y_train = np_utils.to_categorical(y_train)
    y_test = np_utils.to_categorical(y_test)

    scores = []
    scores.append(create_CNN(1, X_train, X_test, y_train, y_test, 7, 27))
    scores.append(create_CNN(1, X_train, X_test, y_train, y_test, 14, 27))
    scores.append(create_CNN(1, X_train, X_test, y_train, y_test, 21, 27))
    scores.append(create_CNN(0, X_train, X_test, y_train, y_test))
    print(scores)
    plt.xlabel('perc_of_opennes, %')
    plt.ylabel('accuracy')
    perc_opennes = [25, 50, 75, 100]
    h = plt.plot(perc_opennes, scores, 'r', linewidth=4)
    plt.show()
```