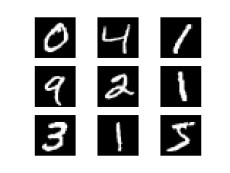
**Тестовые задания на вакансию Разработчик в R&D, Azoft**

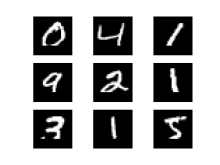
1. **Распознавание частично закрытых рукописных цифр MNIST**

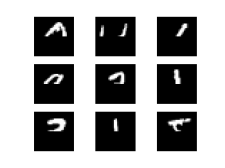
**Дано:** Есть рукописные цифры MNIST



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

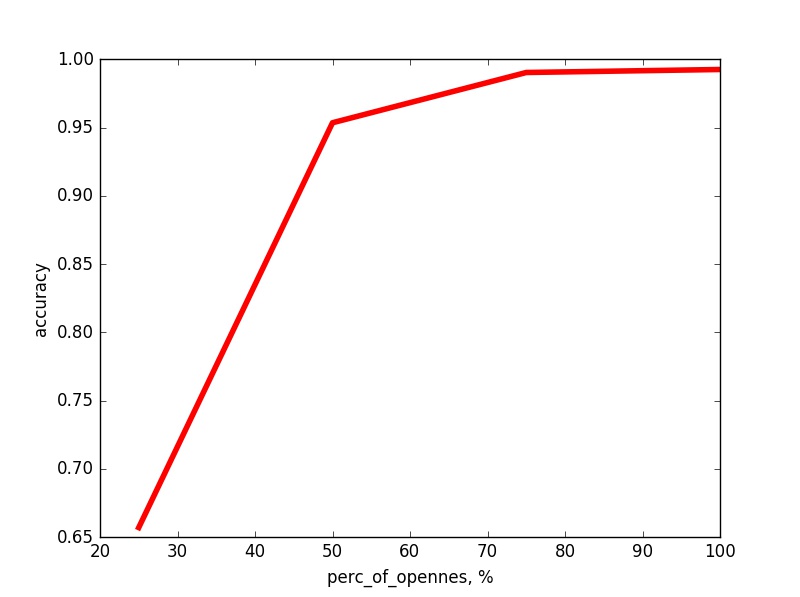
Цифры открыты на 75% (пиксели с y = 21…27 занулены)

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()