Лабораторная работа №3. Реализация сверточной нейронной сети

Данные: В работе предлагается использовать набор данных notMNIST, который состоит из изображений размерностью 28×28 первых 10 букв латинского алфавита (A … J, соответственно). Обучающая выборка содержит порядка 500 тыс. изображений, а тестовая – около 19 тыс.

Данные можно скачать по ссылке: https://commondatastorage.googleapis.com/books1000/notMNIST_large.tar.gz (https://commondatastorage.googleapis.com/books1000/notMNIST_large.tar.gz) (большой набор данных); https://commondatastorage.googleapis.com/books1000/notMNIST_small.tar.gz (https://commondatastorage.googleapis.com/books1000/notMNIST_small.tar.gz) (маленький набор данных); Описание данных на английском языке доступно по ссылке: http://yaroslavvb.blogspot.sg/2011/09/notmnist-dataset.html (http://yaroslavvb.blogspot.sg/2011/09/notmnist-dataset.html)

In [0]:
```
import tensorflow as tf
from scipy.io import loadmat
import numpy as np
from keras.utils import to_categorical
from tensorflow.keras import layers, models, Sequential
```

Задание 1. Реализуйте нейронную сеть с двумя сверточными слоями, и одним полносвязным с нейронами с кусочно-линейной функцией активации. Какова точность построенное модели?

In [0]:
```
def getDataset(file):
    matfile = loadmat(file)
    x = matfile['images'] / 255
    y = matfile['labels']
    x, y = x.T, y.T
    x = x.reshape(x.shape[0], x.shape[1], x.shape[2], 1)
    y = to_categorical(y)
    return x, y
```

In [11]:

```
trainX, trainY = getDataset("/content/drive/My Drive/Collab Dat
print(trainX.shape, trainY.shape)
testX, testY = getDataset("/content/drive/My Drive/Collab Data/
print(testX.shape, testY.shape)
```

```
(529115, 28, 28, 1) (529115, 1)
(18724, 28, 28, 1) (18724, 1)
```

In [14]:

```
model = Sequential([
    layers.Conv2D(64, kernel_size=3, activation='relu', input_s
    layers.Conv2D(32, kernel_size=3, activation='relu'),
    layers.Flatten(),
    layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam', loss='sparse_categorical_crosse

model.fit(trainX, trainY, validation_data=(testX, testY), epoch
```

```
Train on 529115 samples, validate on 18724 samples
Epoch 1/15
529115/529115 [==============================] - 52s 98us/sample -
loss: 0.3813 - acc: 0.8950 - val_loss: 0.1521 - val_acc: 0.9593
Epoch 2/15
529115/529115 [==============================] - 52s 99us/sample -
loss: 0.3055 - acc: 0.9151 - val_loss: 0.1340 - val_acc: 0.9636
Epoch 3/15
529115/529115 [==============================] - 52s 99us/sample -
loss: 0.2730 - acc: 0.9237 - val_loss: 0.1514 - val_acc: 0.9601
Epoch 4/15
529115/529115 [==============================] - 52s 98us/sample -
loss: 0.2502 - acc: 0.9300 - val_loss: 0.1371 - val_acc: 0.9629
Epoch 5/15
529115/529115 [==============================] - 52s 98us/sample -
loss: 0.2318 - acc: 0.9345 - val_loss: 0.1356 - val_acc: 0.9632
Epoch 6/15
529115/529115 [==============================] - 52s 98us/sample -
loss: 0.2173 - acc: 0.9382 - val_loss: 0.1403 - val_acc: 0.9622
Epoch 7/15
529115/529115 [==============================] - 52s 99us/sample -
loss: 0.2056 - acc: 0.9412 - val_loss: 0.1465 - val_acc: 0.9623
Epoch 8/15
529115/529115 [==============================] - 52s 99us/sample -
loss: 0.1951 - acc: 0.9440 - val_loss: 0.1500 - val_acc: 0.9614
Epoch 9/15
529115/529115 [==============================] - 52s 98us/sample -
loss: 0.1859 - acc: 0.9460 - val_loss: 0.1556 - val_acc: 0.9598
Epoch 10/15
529115/529115 [==============================] - 52s 98us/sample -
loss: 0.1784 - acc: 0.9480 - val_loss: 0.1649 - val_acc: 0.9589
Epoch 11/15
```

```
Epoch 11/15
529115/529115 [==============================] - 52s 98us/sample -
loss: 0.1716 - acc: 0.9501 - val_loss: 0.1748 - val_acc: 0.9579
Epoch 12/15
529115/529115 [==============================] - 52s 98us/sample -
loss: 0.1652 - acc: 0.9516 - val_loss: 0.1793 - val_acc: 0.9568
Epoch 13/15
529115/529115 [==============================] - 52s 98us/sample -
loss: 0.1597 - acc: 0.9530 - val_loss: 0.1783 - val_acc: 0.9592
Epoch 14/15
529115/529115 [==============================] - 52s 99us/sample -
loss: 0.1548 - acc: 0.9545 - val_loss: 0.1904 - val_acc: 0.9559
Epoch 15/15

529115/529115 [==============================] - 52s 98us/sample -
loss: 0.1498 - acc: 0.9558 - val_loss: 0.1948 - val_acc: 0.9586
```

Out[14]:
```
<tensorflow.python.keras.callbacks.History at 0x7fddba1af5c0>
```

In [16]:
```
testLoss, testAcc = model.evaluate(testX, testY, verbose=2)
print('Точность на проверочных данных:', testAcc)
```

```
18724/18724 - 1s - loss: 0.1948 - acc: 0.9586
Точность на проверочных данных: 0.9586093
```

Задание 2. Замените один из сверточных слоев на слой, реализующий операцию пулинга (Pooling) с функцией максимума или среднего. Как это повлияло на точность классификатора?

In [17]:
```
modelWithPooling = Sequential([
    layers.Conv2D(64, kernel_size=3, activation='relu', input_s
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(10, activation='softmax')
])

modelWithPooling.compile(optimizer='adam', loss='sparse_categor

modelWithPooling.fit(trainX, trainY, validation_data=(testX, te
```

```
Train on 529115 samples, validate on 18724 samples
Epoch 1/15
529115/529115 [==============================] - 50s 94us/sample -
loss: 0.4351 - acc: 0.8830 - val_loss: 0.1937 - val_acc: 0.9501
Epoch 2/15
529115/529115 [==============================] - 47s 89us/sample -
loss: 0.3723 - acc: 0.8998 - val_loss: 0.1731 - val_acc: 0.9541
Epoch 3/15
529115/529115 [==============================] - 48s 90us/sample -
loss: 0.3494 - acc: 0.9053 - val_loss: 0.1665 - val_acc: 0.9560
Epoch 4/15
```

```
529115/529115 [==============================] - 48s 90us/sample - loss: 0.3350 - acc: 0.9088 - val_loss: 0.1646 - val_acc: 0.9566
Epoch 5/15
529115/529115 [==============================] - 47s 89us/sample - loss: 0.3243 - acc: 0.9116 - val_loss: 0.1615 - val_acc: 0.9584
Epoch 6/15
529115/529115 [==============================] - 47s 90us/sample - loss: 0.3168 - acc: 0.9132 - val_loss: 0.1600 - val_acc: 0.9581
Epoch 7/15
529115/529115 [==============================] - 47s 89us/sample - loss: 0.3101 - acc: 0.9149 - val_loss: 0.1565 - val_acc: 0.9586
Epoch 8/15
529115/529115 [==============================] - 47s 88us/sample - loss: 0.3053 - acc: 0.9159 - val_loss: 0.1600 - val_acc: 0.9576
Epoch 9/15
529115/529115 [==============================] - 47s 89us/sample - loss: 0.3009 - acc: 0.9171 - val_loss: 0.1610 - val_acc: 0.9579
Epoch 10/15
529115/529115 [==============================] - 48s 91us/sample - loss: 0.2972 - acc: 0.9181 - val_loss: 0.1594 - val_acc: 0.9568
Epoch 11/15
529115/529115 [==============================] - 47s 90us/sample - loss: 0.2938 - acc: 0.9187 - val_loss: 0.1602 - val_acc: 0.9564
Epoch 12/15
529115/529115 [==============================] - 47s 89us/sample - loss: 0.2913 - acc: 0.9191 - val_loss: 0.1607 - val_acc: 0.9573
Epoch 13/15
529115/529115 [==============================] - 47s 89us/sample - loss: 0.2890 - acc: 0.9199 - val_loss: 0.1639 - val_acc: 0.9557
Epoch 14/15
529115/529115 [==============================] - 48s 90us/sample - loss: 0.2867 - acc: 0.9205 - val_loss: 0.1625 - val_acc: 0.9575
Epoch 15/15
529115/529115 [==============================] - 47s 89us/sample - loss: 0.2848 - acc: 0.9208 - val_loss: 0.1619 - val_acc: 0.9567
```

Out[17]:

```
<tensorflow.python.keras.callbacks.History at 0x7fddb3eeeb38>
```

In [18]:

```python
testWithPoolingLoss, testWithPoolingAcc = modelWithPooling.eval
print('Точность на проверочных данных с пулингом:', testWithPoo
```

```
18724/18724 - 1s - loss: 0.1619 - acc: 0.9567
Точность на проверочных данных с пулингом: 0.95674
```

Задание 3. Реализуйте классическую архитектуру сверточных сетей LeNet-5 ([http://yann.lecun.com/exdb/lenet/ (http://yann.lecun.com/exdb/lenet/)](http://yann.lecun.com/exdb/lenet/)).

In [19]:

```python
leNet5model = Sequential([
    layers.Conv2D(filters=6, kernel_size=3, activation='relu',
    layers.AveragePooling2D(),
    layers.Conv2D(filters=16, kernel_size=3, activation='relu')
    layers.AveragePooling2D(),
```

```
    layers.Flatten(),
    layers.Dense(units=120, activation='relu'),
    layers.Dense(units=84, activation='relu'),
    layers.Dense(units=10, activation = 'softmax')
])

leNet5model.compile(optimizer='adam', loss='sparse_categorical_

leNet5model.fit(trainX, trainY, validation_data=(testX, testY),
```

```
Train on 529115 samples, validate on 18724 samples
Epoch 1/15
529115/529115 [==============================] - 49s 93us/sample -
loss: 0.4170 - acc: 0.8726 - val_loss: 0.1603 - val_acc: 0.9530
Epoch 2/15
529115/529115 [==============================] - 48s 91us/sample -
loss: 0.3040 - acc: 0.9059 - val_loss: 0.1341 - val_acc: 0.9592
Epoch 3/15
529115/529115 [==============================] - 48s 91us/sample -
loss: 0.2758 - acc: 0.9140 - val_loss: 0.1201 - val_acc: 0.9630
Epoch 4/15
529115/529115 [==============================] - 48s 90us/sample -
loss: 0.2587 - acc: 0.9191 - val_loss: 0.1198 - val_acc: 0.9651
Epoch 5/15
529115/529115 [==============================] - 48s 90us/sample -
loss: 0.2476 - acc: 0.9223 - val_loss: 0.1181 - val_acc: 0.9647
Epoch 6/15
529115/529115 [==============================] - 48s 91us/sample -
loss: 0.2382 - acc: 0.9249 - val_loss: 0.1159 - val_acc: 0.9656
Epoch 7/15
529115/529115 [==============================] - 49s 92us/sample -
loss: 0.2315 - acc: 0.9267 - val_loss: 0.1096 - val_acc: 0.9677
Epoch 8/15
529115/529115 [==============================] - 48s 91us/sample -
loss: 0.2251 - acc: 0.9288 - val_loss: 0.1109 - val_acc: 0.9672
Epoch 9/15
529115/529115 [==============================] - 48s 90us/sample -
loss: 0.2200 - acc: 0.9301 - val_loss: 0.1083 - val_acc: 0.9682
Epoch 10/15
529115/529115 [==============================] - 48s 90us/sample -
loss: 0.2158 - acc: 0.9314 - val_loss: 0.1093 - val_acc: 0.9678
Epoch 11/15
529115/529115 [==============================] - 48s 90us/sample -
loss: 0.2113 - acc: 0.9327 - val_loss: 0.1108 - val_acc: 0.9688
Epoch 12/15
529115/529115 [==============================] - 48s 90us/sample -
loss: 0.2071 - acc: 0.9338 - val_loss: 0.1110 - val_acc: 0.9675
Epoch 13/15
529115/529115 [==============================] - 48s 90us/sample -
loss: 0.2041 - acc: 0.9348 - val_loss: 0.1129 - val_acc: 0.9674
Epoch 14/15
```

```
529115/529115 [==============================] - 49s 92us/sample -
loss: 0.2010 - acc: 0.9355 - val_loss: 0.1109 - val_acc: 0.9682
Epoch 15/15
529115/529115 [==============================] - 48s 90us/sample -
loss: 0.1984 - acc: 0.9363 - val_loss: 0.1137 - val_acc: 0.9668
```

Out[19]:

```
<tensorflow.python.keras.callbacks.History at 0x7fddb3f58860>
```

In [20]:

```
testLeNetLoss, testLeNetAcc = leNet5model.evaluate(testX, testY
print('Точность на проверочных данных с пулингом:', testLeNetAc
```

```
18724/18724 - 1s - loss: 0.1137 - acc: 0.9668
Точность на проверочных данных с пулингом: 0.966834
```