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 (большой набор данных); 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

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

In [1]:

```
import os
import tarfile

data_folder = '../data'

def extract(name):
    path = os.path.join(data_folder, name)

    with tarfile.open(path) as tar:
        tar.extractall(data_folder)
```

In [2]:

```
active_dataset = 'notMNIST_small'
```

In [3]:

```
extract(active_dataset + '.tar.gz')
```

In [3]:

```
import numpy as np
import matplotlib.image as mpimg

def load_data(name, classes, h , w):
    X = []
    y = []

    path = os.path.join(data_folder, name)

    for letter_path, dir_names, file_names in os.walk(path):
        for file_name in file_names:
            try:
                img_path = os.path.join(letter_path, file_name)
                img = mpimg.imread(img_path)
                img = img.reshape(h, w, 1)

                X.append(img)

                letter_class = os.path.basename(letter_path)
                index = classes.index(letter_class)

                y.append(index)
            except:
                pass

    m = len(X)
    X = np.array(X)
    y = np.array(y).reshape(m, 1)

    return X, y
```

In [5]:

```
h = 28
w = 28
n = h * w
classes = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J']
X, y = load_data(active_dataset, classes, h, w)

X.shape, y.shape
```

Out[5]:

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

In [6]:

```
from sklearn.model_selection import train_test_split

X_train, X_dev, y_train, y_dev = train_test_split(X, y)
```

In [8]:

```
from tensorflow import keras

def conv_model():
    model = keras.models.Sequential([
        keras.layers.Conv2D(4, (3, 3), activation='relu', input_shape=(h, w, 1)),
        keras.layers.Conv2D(8, (3, 3), activation='relu'),
        keras.layers.Flatten(),
        keras.layers.Dense(50, activation='relu'),
        keras.layers.Dense(len(classes), activation='softmax')
     ])

    model.compile(optimizer='adam',
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])

    return model
```

In [9]:

```
def train(model, epochs):
    model.fit(X_train, y_train, epochs=epochs)

    return model.evaluate(X_dev, y_dev)
```

In [10]:

```
epochs = 3

train(conv_model(), epochs)
```

```
Train on 14043 samples
Epoch 1/3
14043/14043 [==============================] - 51s 4ms/sample - loss: 0.4944 - accuracy: 0.8609
Epoch 2/3
14043/14043 [==============================] - 50s 4ms/sample - loss: 0.2818 - accuracy: 0.9202
Epoch 3/3
14043/14043 [==============================] - 49s 3ms/sample - loss: 0.2045 - accuracy: 0.9375
4681/4681 [==============================] - 1s 239us/sample - loss: 0.3085 - accuracy: 0.9111
```

Out[10]:

```
[0.30854898106718237, 0.9111301]
```

Задание 2.

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

Как это повлияло на точность классификатора?

In [11]:

```
def conv_max_pool_model():
    model = keras.models.Sequential([
        keras.layers.Conv2D(4, (3, 3), activation='relu', input_shape=(h, w, 1)),
        keras.layers.MaxPool2D(),
        keras.layers.Flatten(),
        keras.layers.Dense(50, activation='relu'),
        keras.layers.Dense(len(classes), activation='softmax')
     ])

    model.compile(optimizer='adam',
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])

    return model
```

In [12]:

```
train(conv_max_pool_model(), epochs)
```

```
Train on 14043 samples
Epoch 1/3
14043/14043 [==============================] - 36s 3ms/sample - loss: 0.6703 - accuracy: 0.8132
Epoch 2/3
14043/14043 [==============================] - 35s 2ms/sample - loss: 0.3783 - accuracy: 0.8954
Epoch 3/3
14043/14043 [==============================] - 33s 2ms/sample - loss: 0.3203 - accuracy: 0.9073
4681/4681 [==============================] - 1s 266us/sample - loss: 0.3559 - accuracy: 0.9015
```

Out[12]:

```
[0.3559267558404152, 0.9015168]
```

Точность не изменилась. Время тренировки уменьшилось.

Задание 3.

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

In [14]:

```
def le_net_5_model():
    model = keras.models.Sequential([
        keras.layers.Conv2D(6, (5, 5), activation='relu', input_shape=(h, w, 1)),
        keras.layers.AveragePooling2D(),
        keras.layers.Conv2D(16, (5, 5), activation='relu'),
        keras.layers.AveragePooling2D(),
        keras.layers.Flatten(),
        keras.layers.Dense(120, activation='relu'),
        keras.layers.Dense(84, activation='relu'),
        keras.layers.Dense(len(classes), activation = 'softmax')
     ])

    model.compile(optimizer='adam',
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])

    return model
```

In [15]:

```
train(le_net_5_model(), epochs)
```

```
Train on 14043 samples
Epoch 1/3
14043/14043 [==============================] - 50s 4ms/sample - loss: 0.7261 - accuracy: 0.7895
Epoch 2/3
14043/14043 [==============================] - 44s 3ms/sample - loss: 0.3969 - accuracy: 0.8848
```

```
Epoch 3/3
14043/14043 [==============================] - 48s 3ms/sample - loss: 0.3162 - accuracy: 0.9057
4681/4681 [==============================] - 1s 266us/sample - loss: 0.3105 - accuracy: 0.9037
```

Out[15]:

```
[0.31053365467450883, 0.9036531]
```

Задание 4.

Сравните максимальные точности моделей, построенных в лабораторных работах 1-3. Как можно объяснить полученные различия?

1. Задача для logistic regression оказалась слишком сложной.
2. Fully connected neural network справилась с задачей довольно неплохо, т к изображения представлены в grayscale и их размер очень мал.
3. С добавлением convolutional layers серьёзного прироста не появилось.