Практическая работа № 7-8

Сверточные нейронные сети (СНС)

Задание 1

а) Обучить представленную СНС в пояснении 1 на встроенном в keras наборе данных Fashion-MNIST. Проанализировать результат, отчет представить в письменном виде с графиками обучения.
б) Доработать и протестировать сеть на обученных весах (на наборе данных Fashion-MNIST или MNIST) на своем изображении (Изображение надо подготовить программно или в редакторе).
в) Доработать, представленную СНС в пояснении 1 таким образом, что бы сеть обучалась на пользовательском наборе данных (набор данных взять на одном из открытых ресурсов: https://www.kaggle.com/datasets; https://public.roboflow.com).

Задание 2

Разработать собственный проект компьютерного зрения, выбор модели за студентом.

Пояснение 1

```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.datasets import mnist        # загрузка набора данных Mnist
from tensorflow import keras
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D
(x_train, y_train), (x_test, y_test) = mnist.load_data()# загрузка набора данных Mnist в массивы для обучения и тестирования
# нормализация входных данных
x_train = x_train / 255
x_test = x_test / 255
y_train_cat = keras.utils.to_categorical(y_train, 10)#Категорирование числовых меток обучающей части набора Mnist
y_test_cat = keras.utils.to_categorical(y_test, 10) #Категорирование числовых меток тестовой части набора Mnist

# модель СНС
model = keras.Sequential([
    Conv2D(32, (3,3), padding='same', activation='relu', input_shape=(28, 28, 1)),
    MaxPooling2D((2, 2), strides=2),
    Conv2D(64, (3,3), padding='same', activation='relu'),
    MaxPooling2D((2, 2), strides=2),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(10,  activation='softmax')
])
#print(model.summary()) # посмотреть параметры СНС
x_train = np.expand_dims(x_train, axis=3)
x_test = np.expand_dims(x_test, axis=3)
```

```
#print( x_train.shape )# Посмотреть входной вектор
#Сборка модели СНС
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
#Обучние модель СНС
his = model.fit(x_train, y_train_cat, batch_size=32, epochs=5, validation_split=0.2)
model.evaluate(x_test, y_test_cat)
```

Функция mnist.load_data() используется для загрузки набора данных MNIST, делит набор данных на обучающий и тестовый в пропорции 50/50.

При вызове mnist.load_data(), функция возвращает кортеж из двух элементов. Первый элемент - это кортеж из двух списков, где каждый список содержит массивы изображений и соответствующих меток для обучающего и тестового наборов данных. Второй элемент - это кортеж из двух массивов, содержащих изображения и метки для валидационного набора данных.

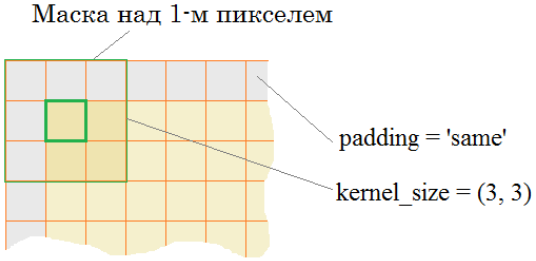
Функция keras.utils.to_categorical используется в библиотеке Keras для преобразования числовых меток в одномерные векторы категорий. В контексте задачи классификации, где каждое входное изображение относится к одной из N категорий (в нашем случае N=10, так как в наборе данных MNIST содержатся цифры от 0 до 9), to_categorical преобразует вектор меток в вектор, где каждый элемент представляет собой один из N возможных классов.

В нашем примере, y_train_cat = keras.utils.to_categorical(y_train, 10), ы преобразуем вектор меток y_train, который содержит числовые значения от 0 до 9, соответствующие каждой цифре, в вектор из 10 категорий. Каждый элемент в этом векторе представляет собой один из классов (цифр), и значение этого элемента равно 1, если соответствующая цифра является целевым классом, и 0 в противном случае.

Таким образом, если у вас есть вектор меток [1, 2, 3, 4], после применения to_categorical вы получите вектор категорий [[1, 0, 0, 0], [0, 1, 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]].

```
model = keras.Sequential([
    Conv2D(32, (3,3), padding='same', activation='relu', input_shape=(28, 28, 1)),
...
])
```

Здесь указано 32 фильтра с ядрами 3x3 пиксела каждый. Затем, параметр padding=’same’ означает, что выходная карта признаков на каждом канале должна быть той же размерностью, что и исходное изображение, т.е. 28x28 элементов. Фактически, это означает добавление значений на границах двумерных данных (обычно нулей), чтобы центр ядра фильтра мог размещаться над граничными элементами:

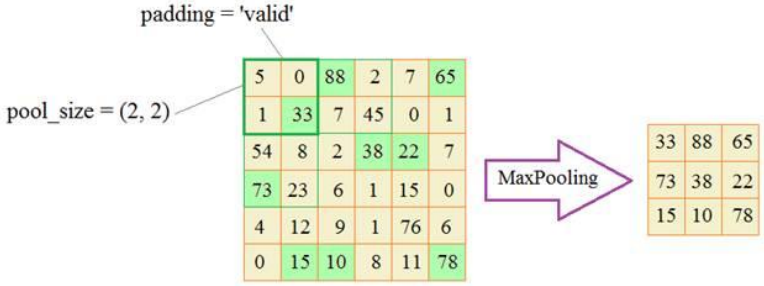


Последние два параметра: функция активации ReLu и формат входных данных в виде изображений 28x28 пикселей с одним цветовым каналом (градации серого).

Следующий слой в соответствии с концепцией СНС должен укрупнять масштаб полученных признаков, для этого чаще всего используется операция MaxPooling:

keras.layers.MaxPooling2D(pool_size=(2, 2), strides=None, padding='valid', data_format=None)

Здесь pool_size – размер окна, в котором выбирается максимальное значение; strides – шаг сканирования по координатам плоскости; padding='valid' – не добавлять нулевых значений на границах (соответственно рамка не смещается за пределы поля признаков); data_format – формат входных данных (об этом поговорим чуть позже).



Добавим в модель операцию (слой) MaxPooling2D:

```
model = keras.Sequential([

    Conv2D(32, (3,3), padding='same', activation='relu', input_shape=(28, 28, 1)),
    MaxPooling2D((2, 2), strides=2),
...
])
```

По аналогии пропишем еще два таких слоя:

```
model = keras.Sequential([

    Conv2D(32, (3,3), padding='same', activation='relu', input_shape=(28, 28, 1)),
    MaxPooling2D((2, 2), strides=2),
    Conv2D(64, (3,3), padding='same', activation='relu'),
    MaxPooling2D((2, 2), strides=2),
...
])
```

Здесь следующий слой свертки содержит уже 64 фильтра, то есть, на выходе будем иметь 64 канала. После операции MaxPooling2D каждая карта признаков уменьшается до размера 7x7 элементов.

Далее, нам нужно вытянуть полученный тензор

7x7x64

в единый вектор. Это выполняется с помощью специального слоя:

keras.layers.Flatten(data_format=None)

И, затем, подать его на полносвязную сеть из 128 нейронов и 10 нейронов выходного слоя. Получаем следующую архитектуру СНС для распознавания рукописных цифр:

```
model = keras.Sequential([

    Conv2D(32, (3,3), padding='same', activation='relu', input_shape=(28, 28, 1)),
    MaxPooling2D((2, 2), strides=2),
    Conv2D(64, (3,3), padding='same', activation='relu'),
    MaxPooling2D((2, 2), strides=2),
    Flatten(),
    Dense(128, activation='relu'),
```

```
    Dense(10,  activation='softmax')
])
```

Давайте выведем структуру этой сети и посмотрим на число весовых коэффициентов в каждом слое:

```
print(model.summary())
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 14, 14, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 7, 7, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 3136)              0
_________________________________________________________________
dense (Dense)                (None, 128)               401536
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1290
=================================================================
Total params: 421,642
Trainable params: 421,642
Non-trainable params: 0
```

Первый слой содержит 320 параметров, второй – 18496, следующий слой полносвязной НС – 401536 и последний – 1290.

Обучим СНС.

Для сверточной НС множества x_tarin и x_test нужно дополнительно подготовить. Дело в том, что на входе такой сети ожидается четырехмерный тензор в формате:

- (batch, channels, rows, cols) – если data_format = 'channels_first';
- (batch, rows, cols, channels) – если data_format = 'channels_last'.

Здесь channels – это каналы на входах сверточных слоев, а параметр data_format по умолчанию равен 'channels_last', что нас вполне устраивает. То есть, наши входные данные должны иметь размерность:

(batch, rows = 28, cols = 28, channels = 1)

Но, сейчас они представлены в виде трехмерного тензора:

(batch, rows = 28, cols = 28)

Нужно к ним добавить еще одно измерение (одну ось) для цветовой компоненты (одноканального изображения). Удобнее всего это сделать с помощью метода expand_dims пакета numpy:

```
x_train = np.expand_dims(x_train, axis=3)
x_test = np.expand_dims(x_test, axis=3)
print( x_train.shape )
```

Теперь входные размерности наших данных соответствуют модели НС.

Параметры компиляции и обучения определим стандартным образом:

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

his = model.fit(x_train, y_train_cat, batch_size=32, epochs=5, validation_split=0.2)
```

И запустим процесс обучения. Точность классификации для обучающего множества и валидации составила 99%. Давайте проверим точность на тестовом множестве:

```
model.evaluate(x_test, y_test_cat)
```

Найденные веса можно сохранить в файл my_model.h5, что бы для предсказаний не обучать сеть заново:

```
model.save('my_model.h5')
```

Что бы загрузить модель:
```
new_model = keras.models.load_model('my_model.h5').
```

Проверим, как работает модель на первом изображении из тестовой части набора данных. Для этого выведем само изображение и передадим его модели для анализа:

```
# Выбираем первое изображение из обучающего набора
first_image = x_train[1]
# Преобразуем изображение в формат, который можно отобразить
first_image = first_image.squeeze()
first_image = first_image.reshape(28, 28)
# Отображаем изображение
plt.imshow(first_image, cmap='gray')
plt.show()
# предсказываем и выводи в консоль предсказанный класс
predictions = new_model.predict(np.array([first_image]))
predicted_digit = np.argmax(predictions)
print(predicted_digit)
```

np.argmax(predictions) возвращает индекс максимального значения в векторе предсказаний. Поскольку мы используем to_categorical для кодирования целевых значений, этот индекс будет соответствовать предсказанной цифре.