Лабораторная работа №4а

Анализ качества работы нейронной сети для распознавания моделей одежды в Keras

1. Цель работы: создание и обучение нейронной сети для распознавания образов (графических изображений) и их последующей классификации .

2. Ход работы:

Пример реализован в среде Google Colab.

Рассмотрим задачу классификации на примере распознавания графических образов, содержащих изображение одежды. Каждое изделие представлено картинкой 28x28 пикселей – 1 пиксель описывается одним байтом в градации серого. Размер входного вектора будет равен 28x28=784 байтам.

Прежде чем перейти к созданию нейронной сети в Keras, рассмотрим теоретический материал, который будет использован в коде программы.

Функция активации softmax

Функция softmax() обычно применяется к выходному слою задач множественной классификации. Она гарантирует, что сумма значений на всех выходных нейронах равна 1, а значение каждого выхода в интервале [0..1] является вероятностью каждого класса. Выход с наибольшим значением вероятности будет использоваться как активный, а все остальные – неактивные.

Выходы нейронов последнего слоя могут быть больше 1 (например, в случае использования функции активации RELU), поэтому функция softmax является идеальным выбором на выходном слое, поскольку она сжимает выход между 0 и 1. Softmax() принимает в качестве входных данных вектор выходных значений предыдущего слоя, суммирует все элементы вектора, а затем делит полученные числа по отдельности на сумму показателей всех чисел во входном векторе. Таким образом общая сумма становится равной 1.

Функция активации softmax() имеет два основных преимущества по сравнению с другими функциями активации, особенно для задач многоклассовой классификации: первое преимущество заключается в том, что функция принимает вектор в качестве входных данных, а второе преимущество заключается в том, что она выдает выходные данные между 0 и 1.

Оценка качества нейросетевой модели

Для оценки качества моделей и сравнения различных алгоритмов используются *метрики*, а их выбор и анализ – непременная часть работы при реализации нейронной сети.

Но прежед чем говорить о метриках, необходимо ввести важную концепцию для описания этих метрик в терминах ошибок классификации — confusion matrix (матрица ошибок).

Допустим, что у нас есть два класса и алгоритм, предсказывающий принадлежность каждого объекта одному из классов, тогда матрица ошибок классификации будет выглядеть следующим образом:

	$y_{истинное} = 1$	$y_{истинное} = 0$
$y_{нейросети} = 1$	True Positive (TP)	False Positive (FP)
$y_{нейросети} = 0$	False Negative (FN)	True Negative (TN)

Таким образом, ошибки классификации бывают двух видов: False Negative (FN) и False Positive (FP).

Интуитивно понятной, очевидной и самой простой метрикой является accuracy – доля правильных ответов алгоритма:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Эта метрика бесполезна в задачах с неравными классами. Но для задачи и начального обучения её применение будет оправдано.

Описание программы

Импортируем необходимые библиотеки и набор данных из корпуса данных mist (набор fashion_mnist – модели одежды):

```
from tensorflow.keras.datasets import fashion_mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras import utils
from tensorflow.keras.preprocessing import image
from google.colab import files

import numpy as np
```

```
import matplotlib.pyplot as plt
from PIL import Image
%matplotlib inline
```

Подготовка данных для обучения сети

Загружаем из Keras набор данных используемый для обучения и проверки:

`x_train-` переменная в которой будут записаны изображения для обучения

`y_train -` переменная в которой будут записаны правильные ответы

```
# В Keras встроены средства работы с популярными наборами данных
# (x_train, y_train) - набор данных для обучения
# (x_test, y_test) - набор данных для тестирования
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()
```

```
32768/29515 [================================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26427392/26421880 [==============================] - 1s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
8192/5148 [===============================================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4423680/4422102 [==============================] - 0s 0us/step
```

Создаем список с названиями классов определяемых объектов (всего в этом наборе их 10):

```
classes = ['футболка', 'брюки', 'свитер', 'платье', 'пальто', 'туфли', 'ру
башка', 'кроссовки', 'сумка', 'ботинки']
```

Просмотрим примеры изображений. Для этого построим график используя библиотеку matplotlib где выведем из заданного диапазона картинки в ячейке размером 10x10:

```
plt.figure(figsize=(10,10))
for i in range(100,150):
    plt.subplot(5,10,i-100+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(x_train[i], cmap=plt.cm.binary)
    plt.xlabel(classes[y_train[i]])
```

Преобразование размерности данных в наборе

Так как изображения находятся в формате 2D, то необходимо графические изображения преобразовать, в формат, подходящий для входа нейронной сети. Для обучения `x_train` будет использовано 60000 изображений в каждой по 784 пикселя, для контрольного множества – 10 000:

```
x_train = x_train.reshape(60000, 784)
x_test = x_test.reshape(10000, 784)
```

Нормирование (или масштабирование) данных

Преобразуем данные для удобства обучения нейронной сети, чтобы на входе они были в диапазоне от 0 до 1 для этого делим интенсивность каждого пикселя в изображении на 255:

```
# Векторизованные операции
# Применяются к каждому элементу массива отдельно
x_train = x_train / 255
x_test = x_test / 255
```

#Выведем правильный ответ для 0-го элемента обучающего множества

```
n = 0
print(y_train[n])
```

```
9
```

Преобразуем метки в формат one hot encoding

Для корректного обучения нейронной сети нужно преобразовать данные правильных ответов из номеров классов в **one hot encoding** с помощью метода `utils.to_categorical`

```
y_train = utils.to_categorical(y_train, 10)
y_test = utils.to_categorical(y_test, 10)
```

Пример ответ в формате one hot encoding:

```
print(y_train[n])
```

```
print(y_train[n])
```

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
```

Создание нейронной сети

Используем модель класса `Sequential`, в которой слои идут последовательно друг за другом:

```
#создаем объект класса Sequential - нейронная сеть
model = Sequential()
```

Указываем, что в нейросети будет входной полносвязный слой,содержащий 800 нейронов, у каждого нейрона 784 входа. В качестве функции активации на входном слое используется функция RELU:

```
model.add(Dense(800, input_dim=784, activation="relu"))
```

Выходной полносвязный слой, 10 нейронов (по количеству классов одежды). В качестве функции активации на выходном слое используем функцию активации softmax:

```
model.add(Dense(10, activation="softmax"))
```

Компиляция нейронной сети

Компиляция нейронной сети выполняется с помощью метода model.compile, указывается функция ошибки – перекрестная энтропия

loss="categorical_crossentropy". Задается тип оптимизатора (т.е. функция, осуществляющая подстройку весов и смещений нейронной сети) optimizer="SGD" - стохастический градиентный спуск. Задается метрика проверки качества обучения сети metrics=["accuracy"].:

```
model.compile(loss="categorical_crossentropy", optimizer="SGD", metrics=["
accuracy"])


print(model.summary())
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 800)               628000
_________________________________________________________________
dense_1 (Dense)              (None, 10)                8010
=================================================================
Total params: 636,010
Trainable params: 636,010
Non-trainable params: 0
_________________________________________________________________
None
```

Обучение нейронной сети

Устанавливаем размер выборки, взяв 200 изображений `batch_size=200,` количество обучений `epochs=100,` доля данных обучения, которые будут использоваться в качестве данных проверки `validation_split=0.2,` выводит прогресс обучения сети:

```
history = model.fit(x_train, y_train,
                    batch_size=200,
                    epochs=100,
                    validation_split=0.2,
                    verbose=1)
```

```
Epoch 1/100
240/240 [==============================] - 1s 6ms/step - loss: 1.1895 - accuracy: 0.6601 - val_loss: 0.8419 - val_accuracy: 0.7391
Epoch 2/100
240/240 [==============================] - 1s 5ms/step - loss: 0.7685 - accuracy: 0.7607 - val_loss: 0.6976 - val_accuracy: 0.7765
Epoch 3/100
240/240 [==============================] - 1s 5ms/step - loss: 0.6687 - accuracy: 0.7900 - val_loss: 0.6313 - val_accuracy: 0.7941
Epoch 4/100
240/240 [==============================] - 1s 5ms/step - loss: 0.6145 - accuracy: 0.8059 - val_loss: 0.5909 - val_accuracy: 0.8087
Epoch 5/100
240/240 [==============================] - 1s 5ms/step - loss: 0.5791 - accuracy: 0.8153 - val_loss: 0.5644 - val_accuracy: 0.8143
Epoch 6/100
240/240 [==============================] - 1s 5ms/step - loss: 0.5538 - accuracy: 0.8214 - val_loss: 0.5454 - val_accuracy: 0.8170
Epoch 7/100
240/240 [==============================] - 1s 5ms/step - loss: 0.5344 - accuracy: 0.8260 - val_loss: 0.5283 - val_accuracy: 0.8225
Epoch 8/100
240/240 [==============================] - 1s 5ms/step - loss: 0.5184 - accuracy: 0.8309 - val_loss: 0.5158 - val_accuracy: 0.8254
Epoch 9/100
240/240 [==============================] - 1s 5ms/step - loss: 0.5058 - accuracy: 0.8337 - val_loss: 0.5044 - val_accuracy: 0.8298
Epoch 10/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4950 - accuracy: 0.8365 - val_loss: 0.4952 - val_accuracy: 0.8316
Epoch 90/100
240/240 [==============================] - 1s 5ms/step - loss: 0.3252 - accuracy: 0.8866 - val_loss: 0.3594 - val_accuracy: 0.8734
Epoch 91/100
240/240 [==============================] - 1s 5ms/step - loss: 0.3239 - accuracy: 0.8879 - val_loss: 0.3590 - val_accuracy: 0.8752
Epoch 92/100
240/240 [==============================] - 1s 5ms/step - loss: 0.3235 - accuracy: 0.8882 - val_loss: 0.3580 - val_accuracy: 0.8738
Epoch 93/100
240/240 [==============================] - 1s 5ms/step - loss: 0.3224 - accuracy: 0.8884 - val_loss: 0.3592 - val_accuracy: 0.8737
Epoch 94/100
240/240 [==============================] - 1s 5ms/step - loss: 0.3211 - accuracy: 0.8892 - val_loss: 0.3618 - val_accuracy: 0.8737
Epoch 95/100
240/240 [==============================] - 1s 5ms/step - loss: 0.3205 - accuracy: 0.8890 - val_loss: 0.3583 - val_accuracy: 0.8733
Epoch 96/100
240/240 [==============================] - 1s 5ms/step - loss: 0.3198 - accuracy: 0.8894 - val_loss: 0.3569 - val_accuracy: 0.8750
Epoch 97/100
240/240 [==============================] - 1s 5ms/step - loss: 0.3192 - accuracy: 0.8889 - val_loss: 0.3600 - val_accuracy: 0.8735
Epoch 98/100
240/240 [==============================] - 1s 5ms/step - loss: 0.3180 - accuracy: 0.8899 - val_loss: 0.3557 - val_accuracy: 0.8748
Epoch 99/100
240/240 [==============================] - 1s 5ms/step - loss: 0.3174 - accuracy: 0.8892 - val_loss: 0.3544 - val_accuracy: 0.8750
Epoch 100/100
240/240 [==============================] - 1s 5ms/step - loss: 0.3164 - accuracy: 0.8901 - val_loss: 0.3564 - val_accuracy: 0.8746
```

Сохраняем обученную нейронную сеть для последующего использования

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

Оценка качества обучения

Проверка качества работы на наборе данных для тестирования:

```
scores = model.evaluate(x_test, y_test, verbose=1)
```

```
313/313 [==============================] - 1s 2ms/step - loss: 0.3801 - accuracy: 0.8657
```

```
print("Доля верных ответов на тестовых данных, в процентах:", round(scores[1] * 100, 4))
```

```
Доля верных ответов на тестовых данных, в процентах: 86.57
```

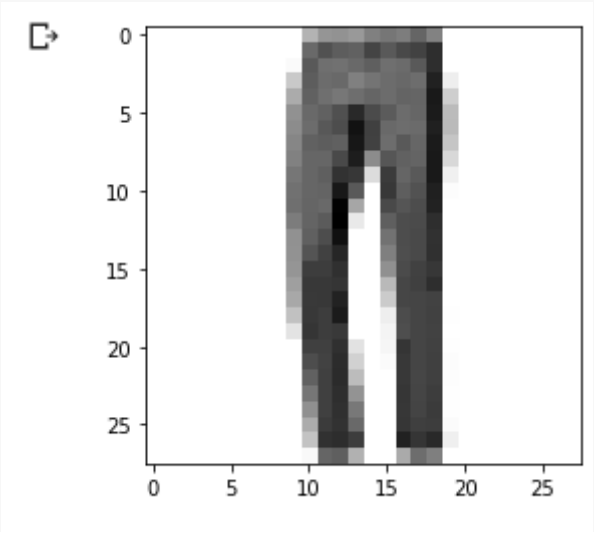
Используем сеть для распознавания предметов одежды

Для примера выберем картинку №496 и выведем её

```
n_rec = 496

plt.imshow(x_test[n_rec].reshape(28, 28), cmap=plt.cm.binary)
plt.show()

#Преобразуем картинку для корректной подачи на вход нейронной сети
```

```
x = x_test[n_rec]
x = np.expand_dims(x, axis=0)

#Запускаем распознавание
prediction = model.predict(x)

#Преобразуем результаты из формата one hot encoding и печатаем
prediction = np.argmax(prediction[0])
print("Сеть распознала номер класса:", prediction)
print("Название класса:", classes[prediction])

#Сравним с правильным ответом
label = np.argmax(y_test[0])
print("Правильный ответ: номер класса:", label)
print("Название класса:", classes[label])
```

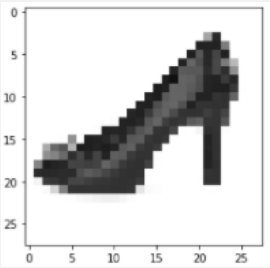
```
313/313 [==============================] - 1s 3ms/step - loss: 0.3792 -
accuracy: 0.8662
Доля верных ответов на тестовых данных, в процентах: 86.62
```



Теперь попробуем подать на сеть изображение, которого не было в обучающем и тестовом наборах. Создадимсамостоятельно или найдем файлы в Сети: 'БРЮКИ ПРОБА.jpg' и 'ТУФЕЛЬКА1 ПРОБА.jpg'

```
#Загружаем свою картинку
files.upload()
#Проверяем загрузку картинки
!ls
img_path = 'ТУФЕЛЬКА1 ПРОБА.jpg'
# можно исользовать этот файл - img_path = 'БРЮКИ1 ПРОБА.jpg'
img = image.load_img(img_path, target_size=(28, 28), color_mode = "grayscale")

#Показываем картинку
plt.imshow(img.convert('RGBA'))
plt.show()
```

```
#Преобразуем картинку в массив
x = image.img_to_array(img)
# Меняем форму матрицы в плоский вектор для подачи на вход нейронной сети
x = x.reshape(1, 784)
# Инвертируем изображение
x = 255 - x
# Нормализуем изображение
x /= 255

#Запускаем распознавание
prediction = model.predict(x)

#Результаты распознавания
prediction
prediction = np.argmax(prediction)
print("Сеть распознала номер класса:", prediction)
print("Название класса:", classes[prediction])

 Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving ТУФЕЛЬКА1 ПРОБА.jpg to ТУФЕЛЬКА1 ПРОБА.jpg
 fashion_mnist_dense.h5  'БРЮКИ1 ПРОБА.jpg'
 sample_data                        'ТУФЕЛЬКА1 ПРОБА.jpg'
```




```
Сеть распознала номер класса: 5

Название класса: туфли
```

Порядок работы

Подготовить собственный набор данных для обучения.
Создать и обучить нейросетевую модель.
Продемотнстрировать результаты обучения (метрики на обучающей и тестовой выборках)
Продемонстрировать работу сети на двух объектах, не использовавшихся в обучающей выборке.
Создать отчет.

Содержание отчета

1. Цель работы
2. Формулировка задачи

3. Описание набора данных для обучения
4. Программная реализация создания нейросетевой модели
5. Результаты работы программы
6. Выводы