Лабораторная работа 6

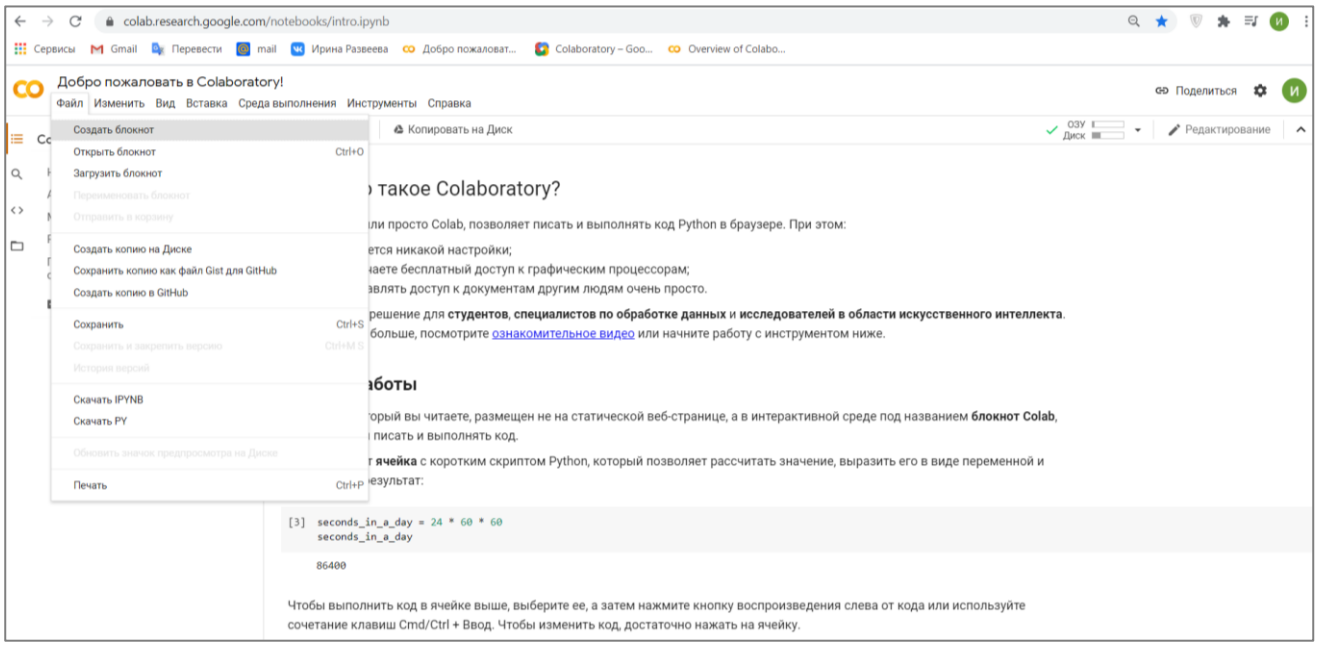
Построение и обучение нейронной сети для распознавания предметы одежды из набора данных Fashion MNIST.

Цель работы: построить нейронную сеть на Python в Google Colaboratory для распознавания предметы одежды из набора данных Fashion MNIST, научиться оценивать влияние гиперпараметров обучения (количество эпох обучения, размер мини-выборки, количество нейронов во входном слое, количество скрытых слоев) на качество обучения нейронной сети.

1. Откройте платформу по ссылке:

https://colab.research.google.com/notebooks/intro.ipynb

2. Перейдя во вкладку Файл выбрать «Создать блокнот».



3. Перед нами пустая рабочая область.

Документ, который вы читаете, размещен не на статической веб-странице, а в интерактивной среде под названием блокнот Colab, позволяющей писать и выполнять код.

В лабораторной работе вы научитесь обучать нейронную сеть распознавать предметы одежды из набора данных Fashion MNIST.

Датасет Fashion MNIST - содержит 70,000 монохромных изображений в 10 категориях. На каждом изображении содержится по одному предмету одежды в низком разрешении (28 на 28 пикселей).

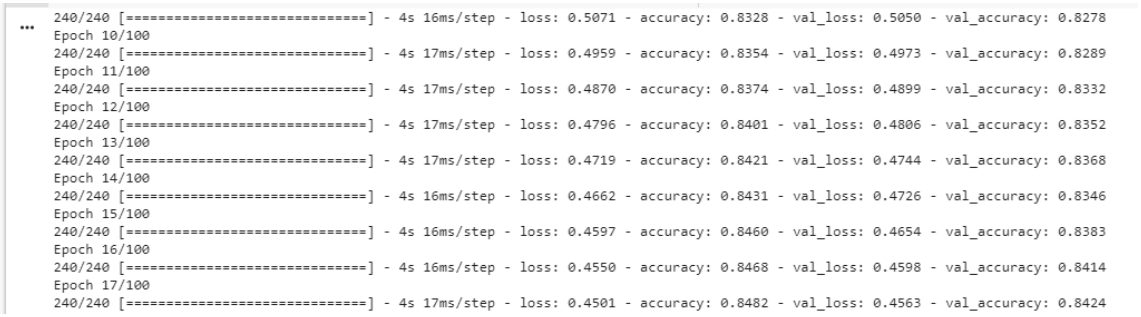
Fashion MNIST предназначен для замены классического датасета MNIST который часто используют как "Hello, World" программ машинного обучения для компьютерного зрения.

Необходимое программное обеспечение:

- библиотека TensorFlow;
- интерфейс для программирования нейросетей Keras.

Код программы:

```
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
import numpy as np

# Загружаем данные
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()

# Список с названиями классов
classes = ['футболка', 'брюки', 'свитер', 'платье', 'пальто',
           'туфли', 'рубашка', 'кроссовки', 'сумка', 'ботинки']

# Преобразование размерности изображений
x_train = x_train.reshape(60000, 784)
x_test = x_test.reshape(10000, 784)
# Нормализация данных
x_train = x_train / 255
x_test = x_test / 255

# Преобразуем метки в категории
y_train = utils.to_categorical(y_train, 10)
y_test = utils.to_categorical(y_test, 10)

# Создаем последовательную модель
model = Sequential()

# Добавляем уровни сети
model.add(Dense(800, input_dim=784, activation="relu"))
model.add(Dense(10, activation="softmax"))

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

print(model.summary())

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

# Оцениваем качество обучения сети на тестовых данных
scores = model.evaluate(x_test, y_test, verbose=1)
print("Доля верных ответов на тестовых данных, в процентах:", round(scores[1]
* 100, 4))
```



```
240/240 [==============================] - 4s 16ms/step - loss: 0.5071 - accuracy: 0.8328 - val_loss: 0.5050 - val_accuracy: 0.8278
Epoch 10/100
240/240 [==============================] - 4s 17ms/step - loss: 0.4959 - accuracy: 0.8354 - val_loss: 0.4973 - val_accuracy: 0.8289
Epoch 11/100
240/240 [==============================] - 4s 17ms/step - loss: 0.4870 - accuracy: 0.8374 - val_loss: 0.4899 - val_accuracy: 0.8332
Epoch 12/100
240/240 [==============================] - 4s 17ms/step - loss: 0.4796 - accuracy: 0.8401 - val_loss: 0.4806 - val_accuracy: 0.8352
Epoch 13/100
240/240 [==============================] - 4s 17ms/step - loss: 0.4719 - accuracy: 0.8421 - val_loss: 0.4744 - val_accuracy: 0.8368
Epoch 14/100
240/240 [==============================] - 4s 16ms/step - loss: 0.4662 - accuracy: 0.8431 - val_loss: 0.4726 - val_accuracy: 0.8346
Epoch 15/100
240/240 [==============================] - 4s 16ms/step - loss: 0.4597 - accuracy: 0.8460 - val_loss: 0.4654 - val_accuracy: 0.8383
Epoch 16/100
240/240 [==============================] - 4s 16ms/step - loss: 0.4550 - accuracy: 0.8468 - val_loss: 0.4598 - val_accuracy: 0.8414
Epoch 17/100
240/240 [==============================] - 4s 17ms/step - loss: 0.4501 - accuracy: 0.8482 - val_loss: 0.4563 - val_accuracy: 0.8424
```

В примере видно, что на начальных эпохах доля правильных ответов на проверочной выборке быстро увеличивается с 0.7548 на первой эпохе до 0.8294 на десятой эпохе. Но ближе к окончанию обучения доля правильных ответов на проверочной выборке почти не меняется. На 97-й эпохе она составляет 0.8764, затем снижается до 0.8740 (эпоха 98) и 0.8758 (эпоха 99), а после этого незначительно увеличивается до 0.8768 на сотой эпохе. Это означает, что мы близки к переобучению и обучение необходимо останавливать.

Имея обученную нейросеть, следует разобраться с гиперпараметрами обучения.

Мы попытаемся улучшить качество обучения сети путем изменения гиперпараметров: количество эпох обучения, размер мини-выборки, количество нейронов во входном слое, количество скрытых слоев. Для этого проведем серию экспериментов, в каждом из которых будем менять один из гиперпараметров, и анализировать, как изменилось качество работы сети.

1. Количество эпох обучения. Оценим влияние количества эпох обучения на качество обучения сети. Количество эпох задается в аргументе epochs метода model.fit:

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

Попробуйте обучать сеть в течение 50, 75, 100 и 125 эпох. Выберите количество эпох, при котором самая высокая доля верных ответов нейросети на тестовых данных.

2. Размер мини-выборки. Оценим влияние размера мини-выборки на качество обучения сети. Размер задается в аргументе batch_size метода model.fit:

```
history = model.fit(x_train, y_train,
                    batch_size=200, # Размер мини-выборки
                    epochs=100,
                    validation_split=0.2,
                    verbose=1)
```

Используйте размер мини-выборки 50, 100, 200 и 400. Выберите значение, при котором самая высокая доля верных ответов нейросети на тестовых данных.

3. Количество нейронов входного слоя. Изменяйте количество нейронов во входном слое и оцените, как оно влияет на качество обучения сети. Количество нейронов задается при создании входного слоя:

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

Используйте значения 500, 700, 900, 1200. Выберите значение, при котором самая высокая доля верных ответов нейросети на тестовых данных.

4. Добавляем скрытый слой. Добавим в нашу сеть скрытый слой, чтобы она стала глубокой:

```
model.add(Dense(800, input_dim=784, activation="relu"))
model.add(Dense(600, activation="relu"))# Новый скрытый слой
model.add(Dense(10, activation="softmax"))
```

Попробуйте добавить скрытый слой с разным количеством нейронов: 500, 700, 900 и 1200.

Выберите наиболее подходящее количество нейронов скрытого слоя. Оцените, как изменяется время обучения при добавлении скрытого слоя с разным количеством нейронов.

Самостоятельная работа:

1. Запишите долю верных ответов работы сети после обучения на тестовых данных.

2. Проанализируйте долю верных ответов на проверочном наборе данных в процессе обучения.