**Лабораторная работа 6**

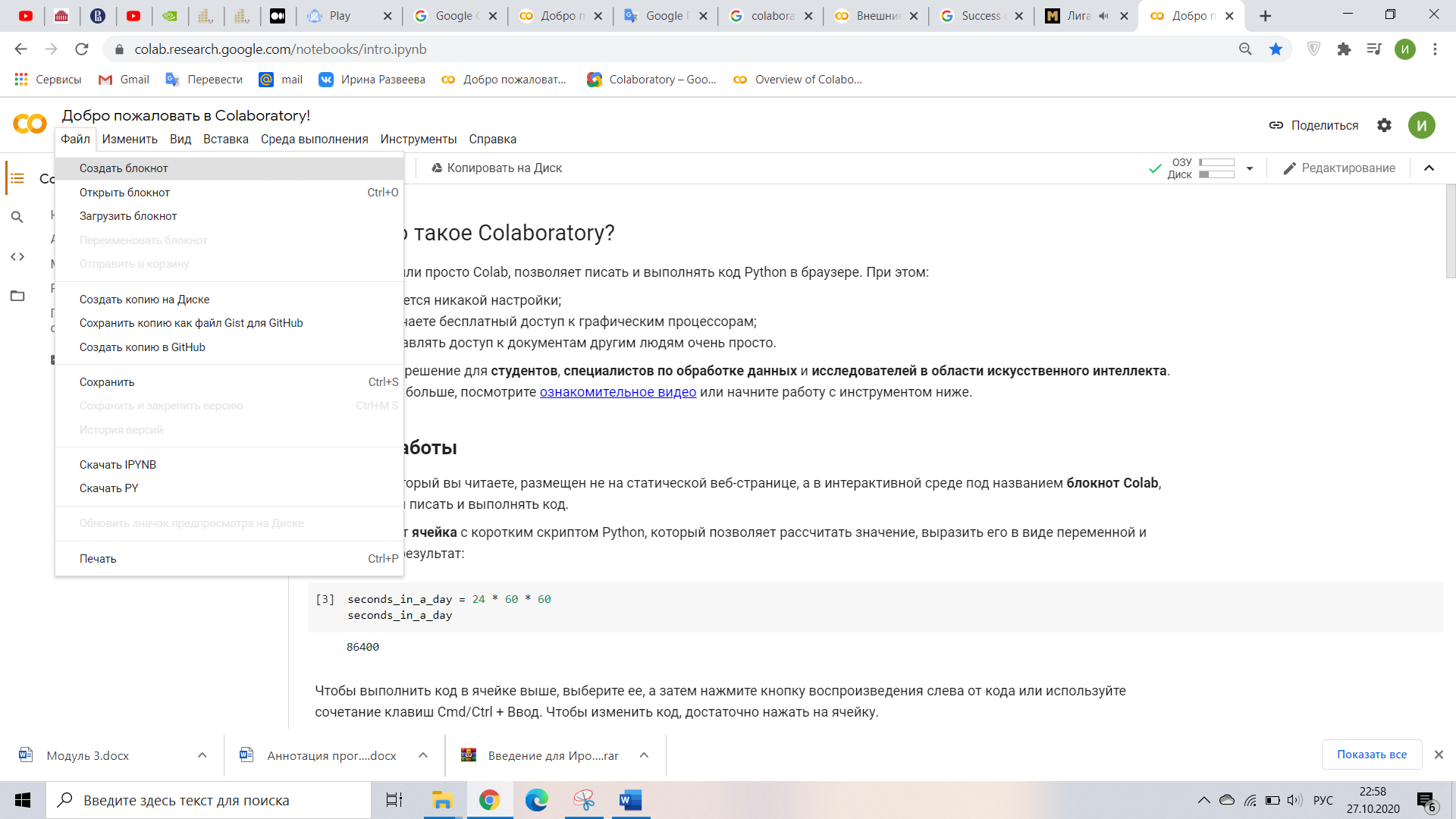
**Построение и обучение нейронной сети для распознавания предметы одежды из набора данных Fashion MNIST.**

**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:

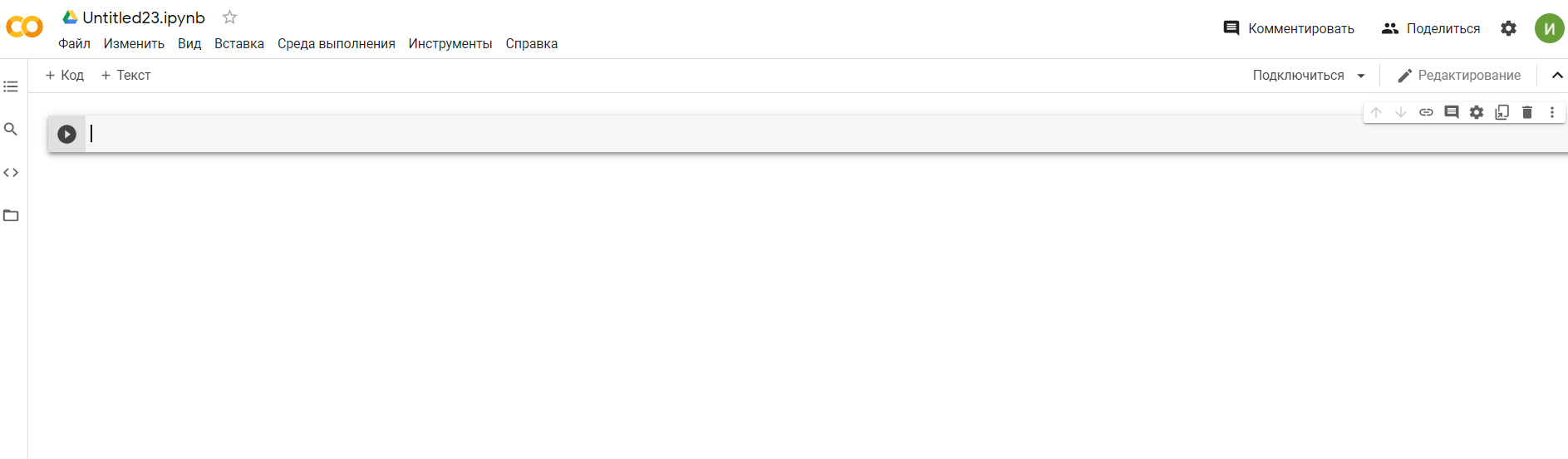
<https://colab.research.google.com/notebooks/intro.ipynb>

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,

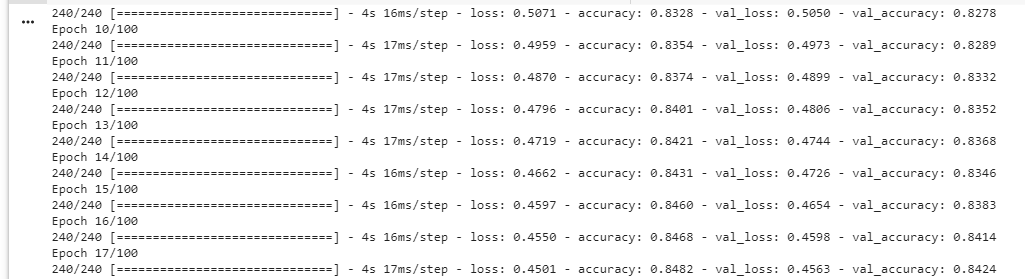
                    validation\_split=0.2,

                    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))



В примере видно, что на начальных эпохах доля правильных ответов на проверочной выборке быстро увеличивается c 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.