## Лабораторная работа 5

## Полносвязная нейронная сеть для распознавания графических образов

## Рассмотрим пример реализации полносвязной нейронной сети, предназначенной для распознавания элементов одежды.

**TensorFlow**-  библиотека с открытым исходным кодом для приложений машинного обучения.  **Keras**, **это** высокоуровневый API (интерфейс прикладного программирования), который может использовать функции **TensorFlow.**

Установка tensorflow:

pip install tensorflow

Для обучения и тестирования нейронной сети используется набор изображений Fashion-MNIST, взятых из статей [Zalando](https://jobs.zalando.com/en/tech/?gh_src=nevh2y1), состоящий из обучающего набора из 60000 примеров и тестового набора из 10000 примеров. Каждый пример представляет собой черно-белое изображение 28x28, связанное с меткой из 10 классов.

Изображения загружаются с помощью метода load\_data:

mnist = keras.datasets.fashion\_mnist

(x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()

**x\_train** – коллекция изображений обучающей выборки;

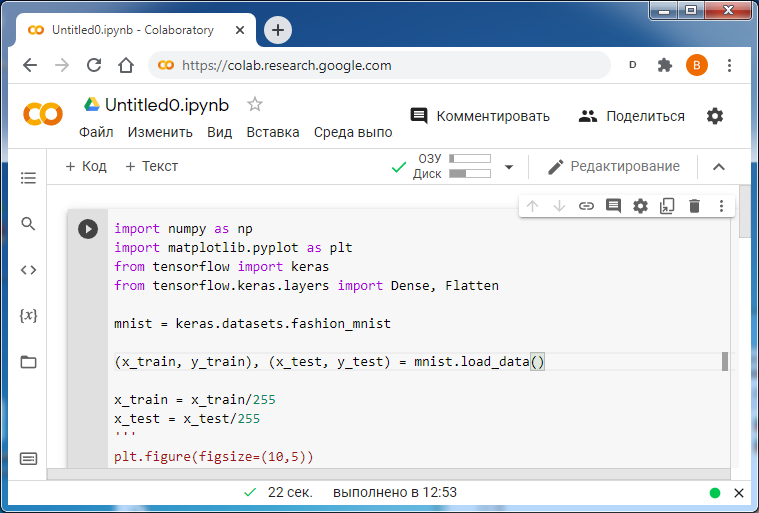
**y\_train** – вектор соответствующих меток (например, если на i-м изображении нарисован свитер, то y\_train[i] = 2);

**x\_test** – коллекция изображений цифр тестовой выборки;

**y\_test** – вектор соответствующих значений цифр для тестовой выборки.

Каждое изображение имеет размер 28x28 пикселей и представлено в градациях серого : (0 – черный цвет, 255- белый)

Выполнять программу будем в **Colaboratorty Google.** Главная особенность «Колаборатории» — бесплатные мощные графические процессоры GPU и TPU, благодаря которым можно заниматься сложными исследованиями в области машинного обучения.



Для просмотра первых 25-и изображений выполните программу:

import matplotlib.pyplot as plt

from tensorflow import keras

mnist = keras.datasets.fashion\_mnist

(x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()

x\_train = x\_train/255

x\_test = x\_test/255

plt.figure(figsize=(10,5))

for i in range(25):

plt.subplot(5,5,i+1)

plt.xticks([])

plt.yticks([])

plt.imshow(x\_train[i],cmap=plt.cm.binary)

plt.show()

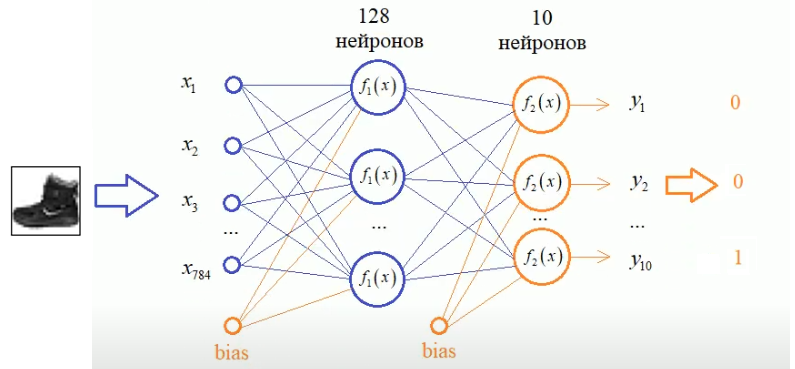
Результат:



Полный список типов элементов одежды из набора Fashion-MNIST:



## Архитектура полносвязной нейронной сети:

****

На первом слое сеть будет содержать 724 входа. Для каждого пикселя отдельный входной элемент. 28x28=784. bias – это смещения, добавляются автоматически.

Для преобразования входной матрицы 28x28 в слой, состоящий из вектора 784 элементов, в Keras предназначен класс:

Flatten(input\_shape=(28,28,1))

Третий параметр равен 1. На один пиксел выделяется 1 байт, хранящий значения от 0 (черный цвет) до 255 (белый цвет).

Скрытый слой содержит 128 нейронов. На данный момент не существует четких рекомендаций по определению количества скрытых слоев и количества нейронов в них. В каждом конкретном случае выбираются опытным путем.

В качестве функции активации для скрытого слоя будем использовать ReLu

Это наиболее часто используемая функция при глубоком обучении. Данная функция возвращает 0, если принимает отрицательный аргумент, в случае положительного аргумента возвращает само число.

Для связывания каждого входа с каждым нейроном скрытого слоя (полносвязная сеть) используется класс

Dense(128,activation=’relu’)

Слой Dense создает связи с входными элементами автоматически.

Выходной слой содержит 10 нейронов. Каждый выходной нейрон соответствует одному из десяти элементов одежды. Для примера на рисунке при подаче на вход ботинка, десятый нейрон (соответствует элементу одежды с меткой 9) должен выдавать 1, остальные нейроны 0.

В качестве функции активации нейронов выходного слоя используется Softmax. Разница между этой функцией и сигмоидальной в том, что softmax нормализует выходы так, что они суммируются в единицу.



K – количество классов.

*zj* – вход j-го нейрона.

Этот слой также полносвязный, поэтому, для его создания используется класс Dense.

Dense(10,activation=’softmax’)

В Keras модель нейронной сети будет выглядеть так:

model = keras.Sequential ([Flatten(input\_shape=(28,28,1)), Dense(128,activation =’relu’), Dense(10,activation=’softmax’)])

Перед обучением нейронной сети желательно выполнить нормализацию входных данных, т.е. выполнить такое их преобразование, чтобы они принимали значения от 0 до 1.

x\_train = x\_train/255

x\_test = x\_test/255

255 – максимальное значение (белый цвет пикселя).

Также нужно выходным значениям придать правильный формат:

|  |  |
| --- | --- |
| метка | Выходные значения нейронов последнего слоя для обучающей выборки |
| 0 | [1,0,0,0,0,0,0,0,0,0] |
| 1 | [0,1,0,0,0,0,0,0,0,0] |
| 2 | [0,0,1,0,0,0,0,0,0,0] |
| 3 | [0,0,0,1,0,0,0,0,0,0] |
| 4 | [0,0,0,0,1,0,0,0,0,0] |
| … | … |
| 9 | [0,0,0,0,0,0,0,0,0,1] |

Такое преобразование в Keras выполняют следующие команды:

y\_train\_cat = keras.utils.to\_categorical(y\_train,10)

y\_test\_cat = keras.utils.to\_categorical(y\_test,10)

Здесь осуществляется преобразование выходных меток классов элементов одежды в приведенный формат для обучающей и тестирующей выборок.

Компиляция нейронной сети выполняется методом модели compile:

model.compile(optimizer=’adam’,

loss=’categorical\_crossentropy’,

metrics=[‘accuracy’])

Здесь

optimizer = ‘adam’ соответствует использованию при обучении нейронной сети оптимизационного метода градиентного спуска.

loss = ‘categorical\_crossentropy’ – один из наиболее популярных критериев качества обучения нейронной сети.

Metrics = [‘accuracy’] – для оценивания качества будем в процессе обучения и тестирования отслеживать процент ошибок, что боле удобно по сравнению со значениями энтропии.

Для запуска процесса обучения нейронной сети используется метод model.fit:

model.fit(x\_train,y\_train\_cat, batch\_size=32, epochs=5, validation\_split=0.2)

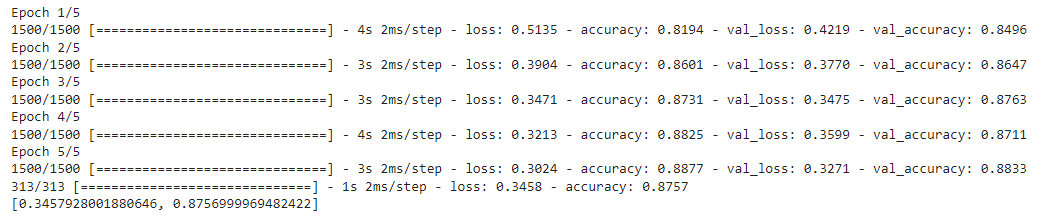
batch\_size (размер партии) = 32 означает, что после каждых 32-х изображений будет производиться корректировка весовых коэффициентов.

validation\_split = 0.2 означает, что 20% из обучающей выборки будет использоваться не для обучения, а для тестирования нейронной сети. Такое распределение обеспечивает репрезентативность как обучающей, так и тестирующей выборки.

Оценивание качества распознавания с использованием тестовой выборки:

Model.evaluate(x\_test,y\_test\_cat)

Результат работы программы:



Точность распознавания составила 88%.

Для более точного распознавания графических образов используются так называемые сверточные нейронные сети.

Добавим в Colab кодовую ячейку и вставим в нее следующий код:

n=355

x = np.expand\_dims(x\_test[n],axis=0)

res = model.predict(x)

print(res)

clothes = ['футболка','шорты','свитер', 'платье','плащ', 'сандали','рубашка','кросовки', 'сумка', 'ботинки']

print(f"одежда: {clothes[np.argmax(res)]}")

plt.imshow(x\_test[n],cmap=plt.cm.binary)

plt.show()

Здесь n – произвольный номер изображения.

Т.к. метод нейронной сети predict (прогноз) ожидает массив изображений, представим наше изображение x\_test[355] как массив из одного изображения.

В полученном массиве выходов

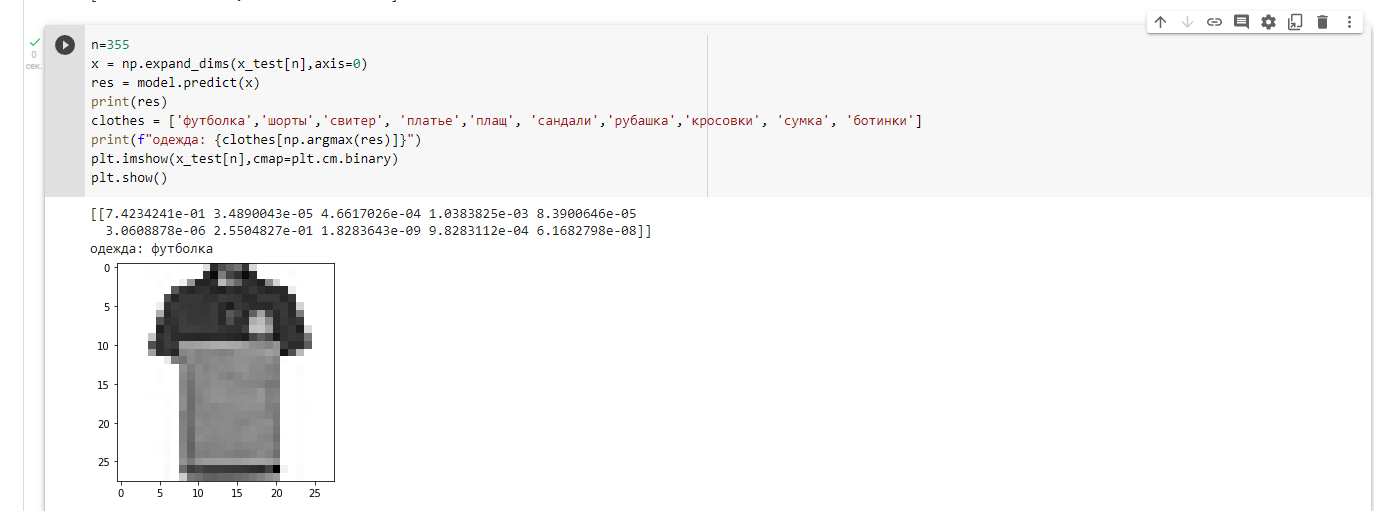


Из полученного массива выходов нейронной сети берем индекс выхода с максимальным значением:



и выводим название прогнозируемого элемента одежды.

Ниже выводим изображение входного изображения.



**Задание**

* Выполните рассмотренные примеры.
* Исследуйте влияние таких параметров, как количество скрытых слоев, количество нейронов в них, количество эпох обучения epochs, batch\_size на качество обучения и распознавания.
* Обучите нейронную сеть распознавать рукописные цифры.

Строку программы

mnist = keras.datasets.fashion\_mnist

замените на

mnist = keras.datasets.mnist

* Сравните качество распознавания элементов одежды и рукописных цифр.