Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего профессионального образования

«Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского»

Институт Информационных Технологий, Математики и Механики

Направление: Прикладная математика и информатика

Магистерская программа: Компьютерные науки и приложения

**ОТЧЕТ**

по лабораторной работе №2

Тема:

**«Разработка полностью связанной нейронной сети »**

Выполнила:

Ягудина Анастасия (гр. 381803м4)

Проверил:

доцент, к.т.н. Кустикова В.Д.

Нижний Новгород

2020

Оглавление

[Постановка задачи 3](#_Toc32151474)

[Описание набора данных 4](#_Toc32151475)

[Метрика качества решения задачи 5](#_Toc32151476)

[Исходный формат хранения данных 5](#_Toc32151477)

[Подготовка данных для нейронной сети 5](#_Toc32151478)

[Пофайловое описание содержимого директории 5](#_Toc32151479)

[Тестовые конфигурации сетей 6](#_Toc32151480)

[Результаты экспериментов 7](#_Toc32151481)

[Анализ результатов 10](#_Toc32151482)

# Постановка задачи

* **Цель работы:**

цель данной работы состоит в том, чтобы получить базовые навыки работы с одной из библиотек глубокого обучения (TensorFlow) на примере полностью связанных нейронных сетей.

* **Задачи:**

1. Выбор библиотеки для выполнения практических работ (TensorFlow);
2. Установка выбранной библиотеки на кластере;
3. Проверка корректности работы библиотеки;
4. Выбор практической задачи компьютерного зрения для выполнения практических работ;
5. Разработка программ/скриптов для подготовки тренировочных и тестовых данных в формате, который обрабатывается данной библиотекой;
6. Разработка нескольких архитектур полностью связанных нейронных сетей (варьируются количество слоев и виды функций активации на каждом слое) в формате, который принимается выбранной библиотекой;
7. Обучение разработанных глубоких моделей;
8. Тестирование обученных глубоких моделей;
9. Публикация разработанных программ/скриптов в личном репозитории на GitHub;
10. Подготовка отчета, содержащего минимальный объем информации по каждому этапу выполнения работы;

# Описание набора данных

Для лабораторных работ был выбран один из представленных на Kaggle набор данных <https://www.kaggle.com/zalando-research/fashionmnist> Fashion MNIST.



*Рис.1. Некоторые элементы набора данных Fashion MNIST.*

Данный набор содержит 70000 монохромных изображений , 60 000 из которых используются для тренировки нейронной сети, а 10 000 - для тестирования. На каждом изображении содержится по одному предмету одежды в низком разрешении. (28х28 пикселей).

Все предметы одежды и обуви из данного набора данных принадлежат ровно одному классу. Всего насчитывается 10 таких классов:



*Рис.2. Классификация набора данных Fashion MNIST.*

Для реализации различных архитектур нейронных сетей была выбрана открытая библиотека глубокого обучения Keras, написанная на языке программирования Python.

# Метрика качества решения задачи

Для оценки качества задачи классификации выбрана метрика "" (точность), показывающая отношение числа правильных предсказаний к числу предсказаний класса для всех изображений:  
где – количество изображений, – значения, выданные сетью при классификации изображений, – истинные значения классов элементов одежды с изображения.

# Исходный формат хранения данных

Каждое изображение составляет 28 пикселей по высоте и 28 пикселей по ширине, в общей сложности 784 пикселей. С каждым пикселем связано значение, указывающее на светлоту или темноту этого пикселя, причем более высокие числа означают более темный цвет. Это значение пикселя является целым числом между 0 и 255. Обучающие и тестовые наборы данных содержат 785 столбцов. Первая колонка состоит из меток класса и представляет собой предмет одежды. Остальные столбцы содержат пиксельные значения соответствующего изображения.

# Подготовка данных для нейронной сети

В качестве входного значения нейронной сети служит одномерный массив длиной 784, так как изображение представляет собой 28х28 пикселей (=784 пикселей всего в изображении), которое преобразуется в одномерный массив. Процесс преобразования 2D-изображения в вектор называется сглаживанием (flattening) и реализуется посредством сглаживающего слоя — flatten-слоя.

# Пофайловое описание содержимого директории

**src** – директория, содержащая разработанный для реализации архитектуры многослойной полностью связанной нейронной сети скрипт.

* Lab\_2.ipynb – описанный выше скрипт, написанный на языке Python 3.

# Тестовые конфигурации сетей

В ходе лабораторной работы были изучены шесть различных кофигураций сетей. Варьировались количество скрытых слоев и функции активации. Схемы конфигураций представлены ниже:

|  |  |  |
| --- | --- | --- |
| Input: Input layer  (28,28)->(None, 784) | ReLu: Dense  (None, 784)->(None, 512) | Softmax: Dense  (None, 512)->(None, 10) |

*Рис 3.* FCNN\_1\_ReLu.

|  |  |  |
| --- | --- | --- |
| Input: Input layer  (28,28)->(None, 784) | Elu: Dense  (None, 784)->(None, 512) | Softmax: Dense  (None, 512)->(None, 10) |

*Рис 4.* FCNN\_2\_Elu.

|  |  |  |
| --- | --- | --- |
| Input: Input layer  (28,28)->(None, 784) | Sigmoid: Dense  (None, 784)->(None, 512) | Softmax: Dense  (None, 512)->(None, 10) |

*Рис 5.* FCNN\_3\_Sig.

|  |  |  |  |
| --- | --- | --- | --- |
| Input: Input layer  (28,28)->(None, 784) | ReLu: Dense  (None, 784)->(None, 512) | ReLu: Dense  (None, 512)->(None, 512) | Softmax: Dense  (None, 512)->(None, 10) |

*Рис 6.* FCNN\_4\_ReLu\_ReLu.

|  |  |  |  |
| --- | --- | --- | --- |
| Input: Input layer  (28,28)->(None, 784) | ELu: Dense  (None, 784)->(None, 512) | ELu: Dense  (None, 512)->(None, 512) | Softmax: Dense  (None, 512)->(None, 10) |

*Рис 7.* FCNN\_5\_Elu\_Elu.

|  |  |  |  |
| --- | --- | --- | --- |
| Input: Input layer  (28,28)->(None, 784) | Sigmoid: Dense  (None, 784)->(None, 512) | Sigmoid: Dense  (None, 512)->(None, 512) | Softmax: Dense  (None, 512)->(None, 10) |

*Рис 8.* FCNN\_6\_Sig\_Sig.

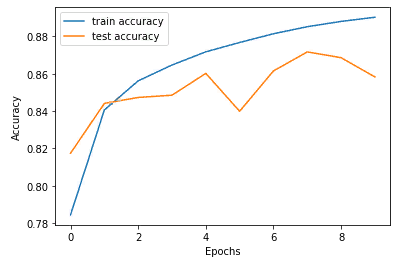
# Результаты экспериментов

**Входные данные:**

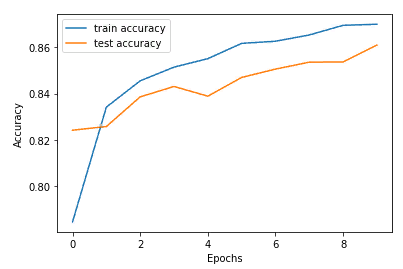
* Train\_images = (60000, 784)
* Train\_labels = (60000, 10)
* Test\_images = (10000, 784)
* Test\_labels = (10000, 10)
* loss = sparse\_categorical\_crossentropy
* optimizer = SGD
* learning\_rate = 0.1
* batch\_size = 128
* num\_epochs = 10

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model name** | **Time train (sec)** | **Test accuracy** | **Test loss** | **Train accuracy** | **Time test (sec)** |
| FCNN\_1\_ReLu | 405 | 0.8711 | 0.3614 | 0.8937 | 4 |
| FCNN\_2\_Elu | 305 | 0.861 | 0.39107 | 0.8785 | 4.12 |
| FCNN\_3\_Sig | 501 | 0.8322 | 0.46604 | 0.84865 | 5.88 |
| FCNN\_4\_relu\_relu | 610 | 0.8801 | 0.3286 | 0.90625 | 7.2 |
| FCNN\_5\_elu\_elu | 778 | 0.8611 | 0.3789 | 0.8826 | 6.59 |
| FCNN\_6\_sig\_sig | 704 | 0.8301 | 0.47513 | 0.84363 | 5.20 |

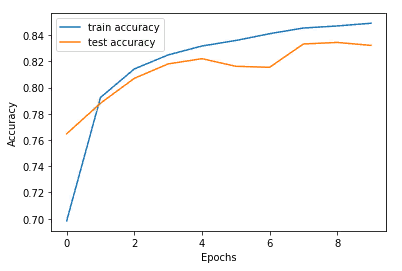
**Графики зависимости точности на тренировочных и тестовых данных от числа эпох**



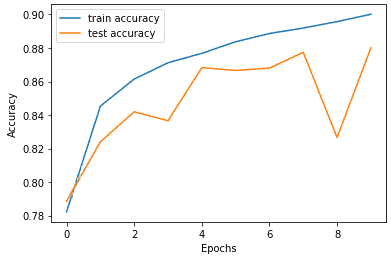
*Рис 9.* FCNN\_1\_ReLu

**

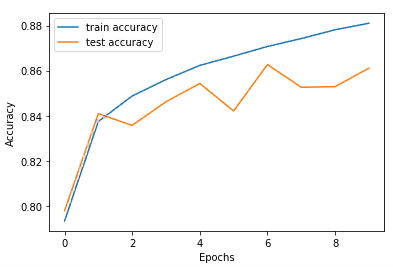
*Рис 10.* FCNN\_2\_ELu



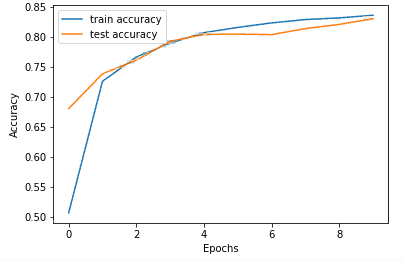
*Рис 11.* FCNN\_3\_Sig



*Рис 12.* FCNN\_4\_ReLu\_ReLu



*Рис 13.* FCNN\_5\_ELu\_Elu



*Рис 14.* FCNN\_6\_Sig\_Sig

# Анализ результатов

Лучший результат показала нейросеть FCNN\_4\_ Relu\_Relu - 0.90625. Дольше всех обучалась нейросеть FCNN\_5\_ Elu\_Elu- 778 sec. Во всех экспериментах наблюдалось переобучение.