МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования   
**«Национальный исследовательский   
Нижегородский государственный университет им. Н.И. Лобачевского»**

**(ННГУ)**

**Институт информационных технологий, математики и механики**

Направление подготовки: «Прикладная математика и информатика»

Магистерская программа: «Вычислительные методы и суперкомпьютерные технологии»

Образовательный курс «Глубокое обучение»

**ОТЧЕТ**

по лабораторной работе №2

**Разработка полностью связанной нейронной сети**

**Выполнил:**

студент группы 381703-3м

Жариков Алексей

Юфин Илья

Веретельников Сергей

Провидохин Николай

Нижний Новгород

2018

Содержание

[Цели 3](#_Toc530326965)

[Задачи 4](#_Toc530326966)

[Решаемая задача 5](#_Toc530326967)

[Выбор библиотеки 6](#_Toc530326968)

[Метрика качества решения задачи 6](#_Toc530326969)

[Тренировочные и тестовые наборы данных 6](#_Toc530326970)

[Конфигурации нейронных сетей 7](#_Toc530326971)

[Описание содержимого директории src 8](#_Toc530326972)

[Результаты экспериментов 8](#_Toc530326973)

[Анализ результатов 9](#_Toc530326974)

[Выводы 10](#_Toc530326975)

[Литература 11](#_Toc530326976)

# Цели

1. ***Цель*** настоящей работы состоит в том, чтобы получить базовые навыки работы с одной из библиотек глубокого обучения (Caffe, Torch, TensorFlow, MXNet или какая-либо другая библиотека на выбор студента) на примере полностью связанных нейронных сетей.

# Задачи

1. Выполнение практической работы предполагает решение ***следующих задач***:
2. Выбор библиотеки для выполнения практических работ курса.
3. Установка выбранной библиотеки на кластере (параметры аутентификации и инструкция по работе с кластером выложена в отдельной задаче в системе redmine).
4. Проверка корректности установки библиотеки. Разработка и запуск тестового примера сети, соответствующей логистической регрессии, для решения задачи классификации рукописных цифр набора данных MNIST (пример разобран в лекционных материалах).
5. Выбор практической задачи компьютерного зрения для выполнения практических работ.
6. Разработка программ/скриптов для подготовки тренировочных и тестовых данных в формате, который обрабатывается выбранной библиотекой.
7. Разработка нескольких архитектур полностью связанных нейронных сетей (варьируются количество слоев и виды функций активации на каждом слое) в формате, который принимается выбранной библиотекой.
8. Обучение разработанных глубоких моделей.
9. Тестирование обученных глубоких моделей.
10. Публикация разработанных программ/скриптов в репозитории на GitHub.
11. Подготовка отчета, содержащего минимальный объем информации по каждому этапу выполнения работы.

# Решаемая задача

В данной работе рассматривается набор данных Fashion MNIST, который содержит 70 000 изображений в оттенках серого в 10 категориях. На изображениях показаны отдельные предметы одежды с низким разрешением (28 на 28 пикселей):

Получить можно данный набор по ссылке

<https://www.kaggle.com/zalando-research/fashionmnist> ,

но в ходе работы использовался функционал используемой библиотеки глубокого обучения, непосредственно из нее, просто импортировав и загрузив данные

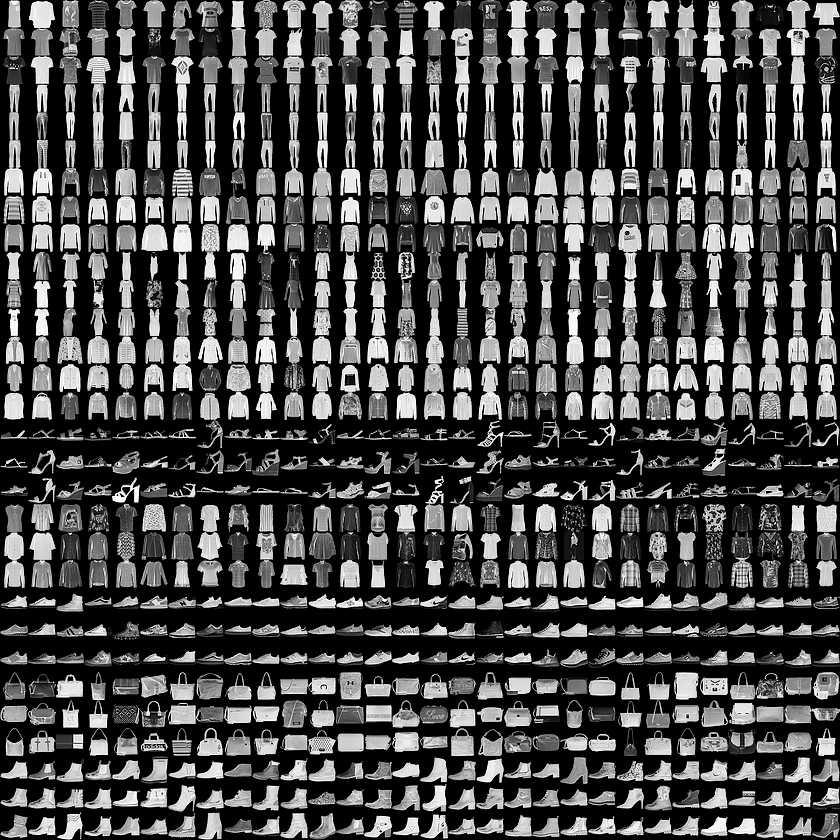


Рисунок 1 Примеры набора данных

# Выбор библиотеки

Для выполнения лабораторных работ выбрана библиотека TensorFlow для языка программирования Python c надстройкой Keras .

На этапе проверки корректности установки библиотеки выполнена разработка и запуск тестового примера сети для решения задачи классификации рукописных цифр набора данных MNIST. Достигнута точность 0.9225.

# Метрика качества решения задачи

В качестве метрики точности решения используется отношение угаданных животных ко всем в тестовой выборке:

# Тренировочные и тестовые наборы данных

# Обучающее множество 60.000 примеров и тестовый набор 10000 примеров. Всего имеется 10 классов различной одежды и обуви.

# Каждый пример обучения и теста назначается одному из следующих классов:

|  |  |  |  |
| --- | --- | --- | --- |
| **Номер** | **Описание** | **Кол.**  **трен** | **Кол.**  **тест** |
| 0 | Футболки / сверху | 6000 | 1000 |
| 1 | Брюки | 6000 | 1000 |
| 2 | Пуловер | 6000 | 1000 |
| 3 | Платье | 6000 | 1000 |
| 4 | Пальто | 6000 | 1000 |
| 5 | Сандали | 6000 | 1000 |
| 6 | Рубашка | 6000 | 1000 |
| 7 | Кроссовки | 6000 | 1000 |
| 8 | Мешок | 6000 | 1000 |
| 9 | Ботильоны | 6000 | 1000 |

Таблица 1 Классы и количество примеров в наборе данных

# Конфигурации нейронных сетей

В данной работе были рассмотрены 11 конфигураций полносвязных нейронных сетей у которых от 1 до 4 скрытых слоев.

Активационная функция на слоях выбирается из следующих:

На выходном слое:

Разработанные программы/скрипты

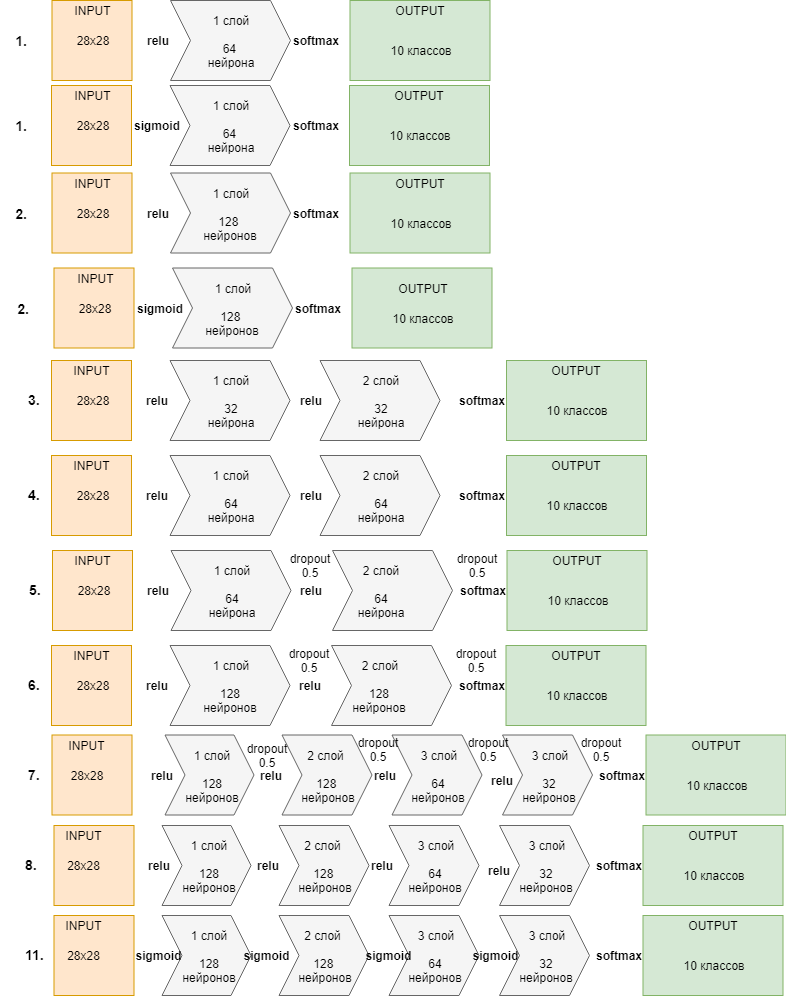
В директории расположен один файл Lab2.ipynb, в котором и находятся рассмотренные конфигурации, описанные ниже

Рисунок 2. Схемы конфигураций сетей

# Результаты экспериментов

Тестовая инфраструктура:

ОП: Python Jupiter cloud colab.research.google.com

В работе рассмотрены несколько конфигураций.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **Количество скрытых слоев** | **Количество нейронов на скрытых слоях** | **Функции активации** | **Кол.**  **эпох** | **Скор.**  **обуч.** | **Результат** | | | |
| Точность на тренировочном множестве | Точность на тестовом множестве | Время обучения, с | Время теста  одного изобрю, 10-5с |
| 1 | 1 | 64 | relu | 20 | 0,01 | 0.9157 | **0.8788** | 67.304 | 8.4600 |
| 2 | 1 | 64 | sigmoid | 20 | 0,01 | 0.8888 | 0.8704 | 67.353 | 8.4891 |
| 3 | 1 | 128 | relu | 20 | 0,01 | 0.9269 | **0.8800** | 67.707 | 8.4487 |
| 4 | 1 | 128 | sigmoid | 20 | 0,01 | 0.8909 | 0.8699 | 67.349 | 8.4483 |
| 5 | 2 | 32-32 | relu- relu | 20 | 0,01 | 0.9061 | **0.8717** | 71.928 | 8.7406 |
| 6 | 2 | 64-64 | relu-relu | 20 | 0,01 | 0.9237 | 0.8765 | 72.931 | 8.779 |
| 7 | 4 | 64-64 | relu-dropout-relu-dropout | 20 | 0,01 | 0.8208 | 0.8505 | 77.859 | 8.831 |
| 8 | 4 | 128-128 | relu-dropout-relu-dropout | 20 | 0,01 | 0.8634 | 0.8688 | 77.693 | 9.247 |
| 9 | 8 | 128-128-  64-32 | relu-dropout-relu-dropout- relu-dropout-relu-dropout | 20 | 0,01 | 0.7695 | 0.8152 | 90.212 | 9.179 |
| 10 | 4 | 128-128-  64-32 | relu-relu- relu-relu | 80 | 0,01 | 0.9827 | **0.8793** | 322.693 | 9.278 |
| 11 | 4 | 128-128-  64-32 | sigmoid - sigmoid - sigmoid - sigmoid | 80 | 0,01 | 0.9523 | **0.8708** | 324.611 | 9.238 |

Таблица 2 Конфигурации и результаты

**Графики точности классификации после обучения на тестовых наборах:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Количество скрытых слоев** | **Количество нейронов на скрытых слоях** | **Функции активации** | Точность на тестовом множестве | **График** |
| 1 | 1 | 64 | relu | **0.8788** |  |
| 2 | 1 | 128 | relu | **0.8800** |  |
| 3 | 2 | 32-32 | relu- relu | **0.8717** |  |
| 4 | 4 | 128-128-  64-32 | relu-relu- relu-relu | **0.8793** |  |

Таблица 3 Графики точностей на тестовых и обучающем наборах

**Матрица расхождения:**

Для каждой конфигурации строилась матрица расхождений, где мы получаем картину, на какие классы сеть дает верные\неверные ответы.

По итогу можно увидеть, что классификация футболок, пальто и рубашки являются основной причиной ошибок сети, что в общем-то и логично, так как образы их похожи.

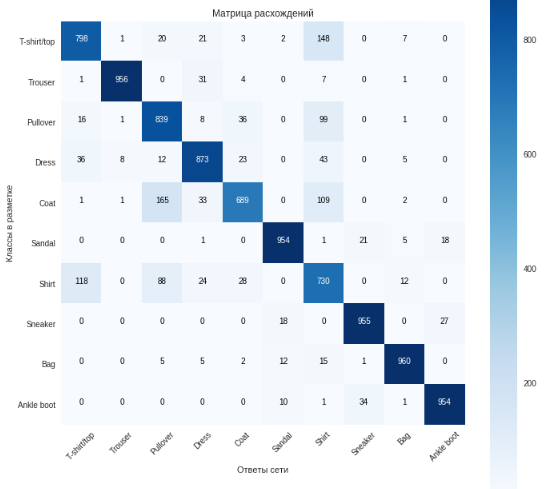


Рисунок 3 Матрица расхождения для 11-ой конфигурации

# Анализ результатов

Нейронные сети с функцией активации ReLU показывают результат лучше, чем нейронные сети с функцией сигмоидальной функцией активации.

Рассмотрим функцию активации, которая представляется суммой нескольких логистических сигмоидов:

Построенную сумму можно приблизить интегралом:

Приведенный ряд сигмоидальных функций более выразителен и может быть приближен .

|  |  |
| --- | --- |
| Рисунок 4 График сигмоиды | Рисунок 5 График ReLU |

Указанный логарифм похож на ReLU.

Преимущества ReLU:

Вычисление сигмоиды и гиперболического тангенса требует ресурсоёмких операций, таких как возведение в степень, в то время как ReLU не подвержен насыщению.

1. Применение ReLU существенно повышает скорость стохастического градиентного спуска по сравнению с сигмоидой и гиперболическим тангенсом. Это обусловлено линейным характером и отсутствием насыщения данной функции.

Недостатки ReLU:

К сожалению, ReLU не всегда достаточно надёжны и в процессе обучения могут выходить из строя. Например, большой градиент, проходящий через ReLU, может привести к такому обновлению весов, что данный нейрон никогда больше не активируется. Если это произойдет, то, начиная с данного момента, градиент, проходящий через этот нейрон, всегда будет равен нулю. Соответственно, данный нейрон будет необратимо выведен из строя. Например, при слишком большой скорости обучения, может оказаться, что до 40% ReLU никогда не активируются. Эта проблема решается посредством выбора надлежащей скорости обучения.

# Выводы

Наилучший результат был получен на нейронной сети с конфигурацией №3. В ходе экспериментов было установлено, что нейронные сети с функцией активации relu показывают более точные результаты. Так же увеличить точность помогло увеличение нейронов на всех слоях. Однако, точность увеличилась не сильно, а время работы более чем в 3 раза.