yМинистерство 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 №3

Тема:

**«Разработка сверточной нейронной сети »**

Выполнила:

Ягудина Анастасия (гр. 381803м4)

Проверил:

доцент, к.т.н. Кустикова В.Д.

Нижний Новгород

2020

Оглавление

[Постановка задачи 3](#_Toc32685316)

[Пофайловое описание содержимого директории 4](#_Toc32685317)

[Тестовые конфигурации сетей 4](#_Toc32685318)

[Результаты экспериментов 5](#_Toc32685319)

[Анализ результатов 9](#_Toc32685320)

# Постановка задачи

* **Цель работы:**

Цель настоящей работы состоит в том, чтобы построить архитектуру сверточной нейронной сети, которая позволяет решать поставленную задачу с более высокими показателями качества.

* **Задачи работы:**

1. Разработка нескольких архитертур сверточных нейронный сетей (варьируются количество слоев и виды функций активаций на каждом слое);
2. Обучение разработанных глубоких моделей;
3. Тестирование обученных глубоких моделей;
4. Публикация разработанных программ/скриптов в личном репозитории GitHub;
5. Подготовка отчета .

*Сверточные нейронные сети* – вид нейронных сетей, которые хотя бы на одном из своих слоев в качестве преобразования используют операцию «свертки». *Свертка* – операция, применяемая к двум вещественнозначным функциям. Операция *свертки* обозначается следующим образом:

***S(t) = <x \* w>(t),***

Где ***x(.) –*** вход, а ***w(.)-*** ядро свертки. Выход называют *картой признаков.* В данной работе в качестве входа рассматривается *тензор-* многомерный массив данных, ядро- многомерный массив параметров, которые подбираются посредством обучения. Если на входе имеется двумерное изображение ***I*** и ядро ***K,*** то операция свертки имеет следующий вид:

Свойства функции свертки: *коммутативность, ассоциативность, линейность.*

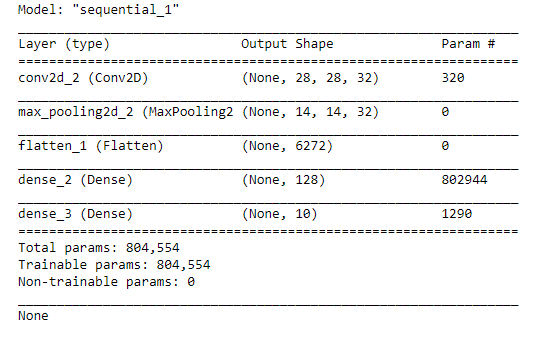
# Пофайловое описание содержимого директории

**src** – директория, содержащая разработанный для реализации архитектуры сверточной нейронной сети скрипт.

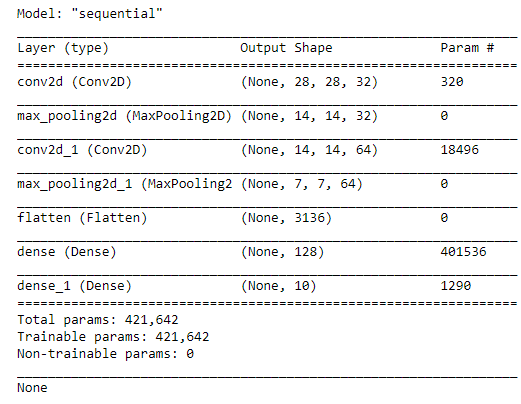
* Lab\_3.ipynb –скрипт, содержащий реализацию архитектуры сверточной нейронной сети, написанный на языке Python 3;
* Lab\_3.docx – отчет по данной работе.

# Тестовые конфигурации сетей

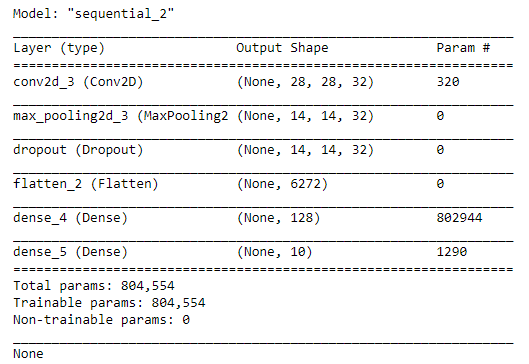
В ходе лабораторной работы были изучены шесть различных конфигураций сетей. Варьировались количество сверточных слоев и функции активации. Схемы конфигураций представлены ниже:



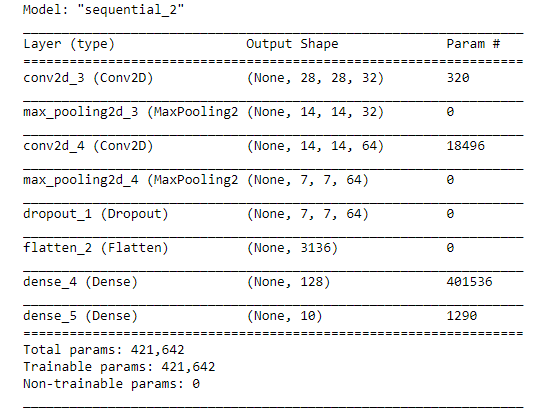
*Рис 1. Архитектура CNN\_4\_ReLu*

**

*Рис 2. Архитектура CNN\_1\_ReLu\_ReLu*



*Рис 3. Архитектура CNN\_6\_ReLu\_Drop*



*Рис 4. Архитектура CNN\_9\_ReLu\_Relu\_Drop.*

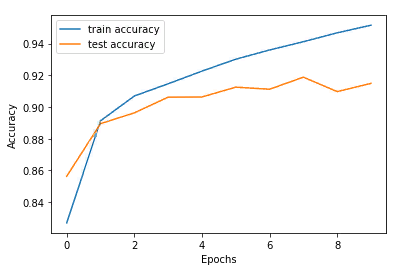
# Результаты экспериментов

**Входные данные:**

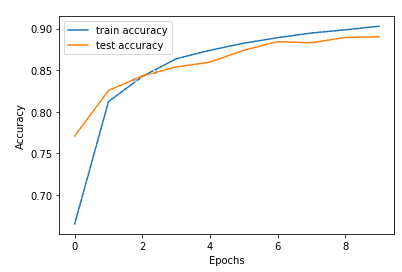
* Train\_images = (60000, (28, 28, 1))
* Train\_labels = (60000, 10)
* Test\_images = (10000, (28, 28, 1))
* Test\_labels = (10000, 10)
* loss = sparse\_categorical\_crossentropy
* optimizer = adam
* batch\_size = 128
* num\_epochs = 10
* learning\_rate = 0,01

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model name** | **Time train (sec)** | **Test accuracy** | **Test loss** | **Train accuracy** | **Time test (sec)** |
| CNN\_1\_ReLu\_ReLu | 7069 | 0.9149 | 0.2407 | 0.9515 | 49.09 |
| CNN\_2\_ Sig\_Sig | 6073 | 0.8902 | 0.3025 | 0.9029 | 42.06 |
| CNN\_3\_Elu\_Elu | 7057 | 0.8745 | 0.3464 | 0.8930 | 48.4 |
| CNN\_4\_Sig | 3376 | 0.7547 | 0.7 | 0.7470 | 36.03 |
| CNN\_5\_ReLu | 2201 | 0.8908 | 0.5505 | 0.9640 | 19.39 |
| CNN\_6\_ReLu\_Drop | 5993 | 0.9 | 0.4 | 0.9401 | 33.18 |
| CNN\_7\_ELu\_Drop | 8338 | 0.8101 | 0.58 | 0.8111 | 63.33 |
| CNN\_8\_ReLu\_ReLu\_Drop | 6773 | 0.9016 | 0.2861 | 0.9183 | 53.30 |
| CNN\_9\_Elu\_Elu\_Drop | 6160 | 0.8369 | 0.48703 | 0.8372 | 37.47 |

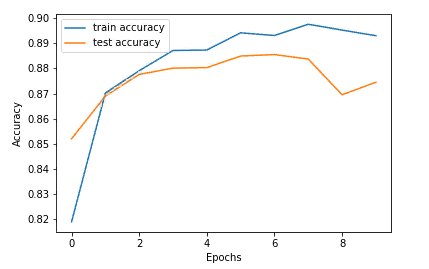
**Графики зависимости точности на тренировочных и тестовых данных от числа эпох**



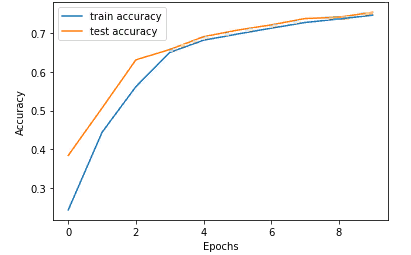
*Рис 1.* CNN\_1\_ReLu\_ReLu



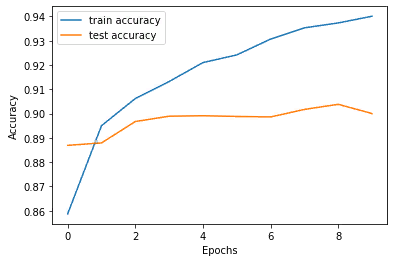
CNN\_2\_Sig\_Sig



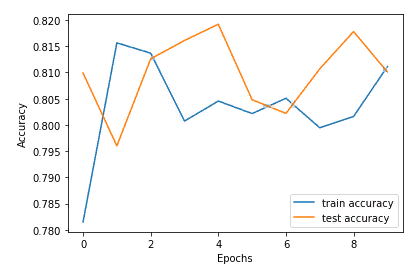
CNN\_3\_ELu\_ELu

**

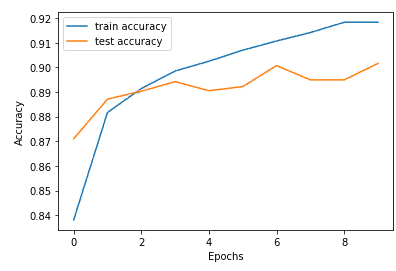
CNN\_4\_Sig



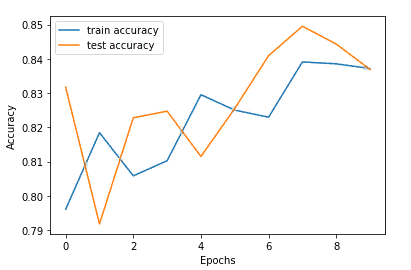
CNN\_6\_Relu\_Drop



CNN\_7\_Elu\_Drop



CNN\_8\_ReLu\_ReLu\_Drop



CNN\_9\_ELu\_Elu\_Drop

# Анализ результатов

Лучший результат показала нейросеть CNN\_1\_ Relu\_Relu - 0.9515 и CNN\_5\_Relu - 0.9640. Дольше всех обучалась нейросеть CNN\_5\_ Elu\_Drop - 8338 sec. Добавление Dropout не сильно улучшило результат.