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 №3

**Разработка сверточных нейронных сетей**

**Выполнили:**

студенты группы 381703-3м

Гладкова Татьяна

Крутоборежская Ирина

Крюкова Полина

Подчищаева Мария

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 построить архитектуру сверточной нейронной сети, которая позволяет решать практическую задачу с высокими показателями качества.

# Задачи

1. Выполнение практической работы предполагает решение ***следующих задач***:
2. 1. Разработка нескольких архитектур сверточных нейронных сетей (варьируются количество слоев и виды функций активации на каждом слое) в формате, который принимается выбранной библиотекой глубокого обучения.
3. 2. Обучение разработанных глубоких моделей.
4. 3. Тестирование обученных глубоких моделей.
5. 4. Публикация разработанных программ/скриптов в репозитории на GitHub.
6. 5. Подготовка отчета, содержащего минимальный объем информации по каждому этапу выполнения работы.

# Решаемая задача

Была выбрана задача бинарной классификации: «кошки» - «собаки». Были использованы картинки из наборов данных https://www.kaggle.com/tongpython/cat-and-dog и [https://www.kaggle.com/c/dogs-vs-cats/data](https://vk.com/away.php?to=https%3A%2F%2Fwww.kaggle.com%2Fc%2Fdogs-vs-cats%2Fdata&cc_key=). Получившийся набор состоит из

35029 изображений.

|  |  |
| --- | --- |
|  |  |
| Рис. 1 Пример изображения из класса «кошки» | Рис. 2 Пример изображения из класса «собаки» |

С помощью скрипта на python данные были преобразованы к размеру 128×128. С помощью скрипта im2rec.py, который входит в библиотеку MXNet, изображения были сконвертированы в формат .rec.

# Выбор библиотеки

Для выполнения лабораторных работ выбрана библиотека MXNet для языка программирования Python.

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

В качестве тренировочной выборки используем тренировочную выборку первого и второго наборов данных, всего 16500 изображений котов и 16505 изображений собак. В качестве тестовой выборки используем тестовую выборку только из первого набора данных, т.к. во втором наборе данных тестовая выборка не разбита на изображения котов и собак. Всего в тестовой выборке 2042 изображения, котов и собак поровну.

# Конфигурации нейронных сетей

В данной работе были рассмотрены четыре конфигурации сверточных нейронных сетей с одним и двумя сверточными слоями. Ядра сверточных слоев выбраны следующих размеров:

• 2х2

• 3х3

В каждом сверточном слое к картинке может применяться по несколько фильтров. Значение фильтров выбирались из следующих:

• 500

• 1000

Так же в сверточных слоях фильтр шел по картинке с некоторым шагом по пикселям. Этот шаг называется stride. И использовались следующие его значения:

• (2, 2)

• (1, 1)

После каждой свертки применялась функция активации:

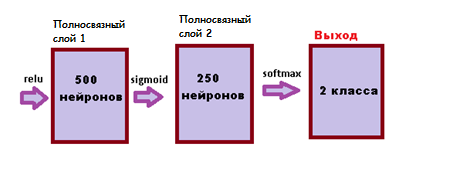
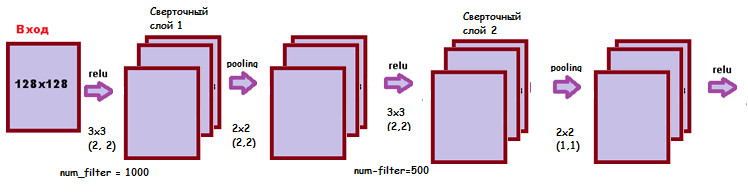
После активации используется операция пространсвенного объединения(pooling):

* Max – pooling

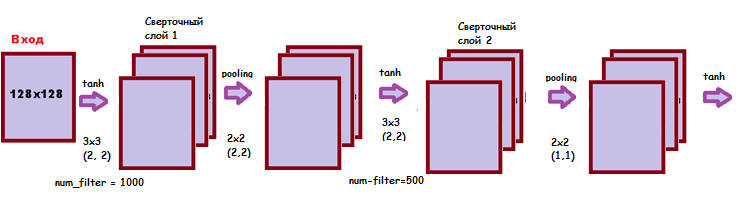
Активационная функция на полносвязных слоях выбирается из следующих:

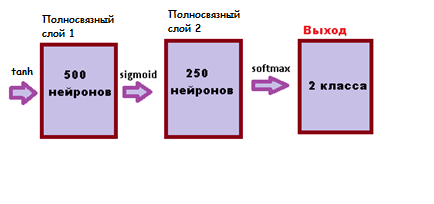
На выходном слое:

1.

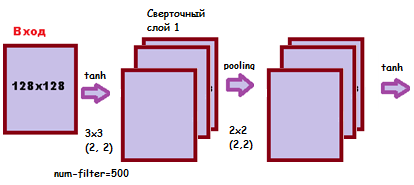


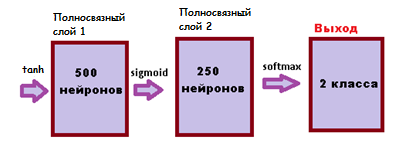
2.



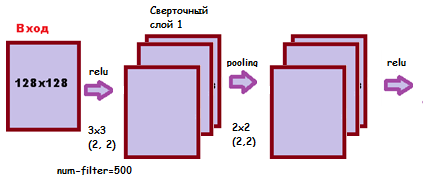


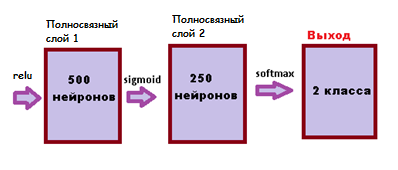
3.





4.





# Разработанные программы/скрипты

В директории расположены четыре конфигурации построенных сверточных нейронных сетей. Соответствия построенных конфигураций и конфигураций в директории:

* CNN\_config\_1.py – первая нейронная сеть с двумя сверточными слоями и с двумя полносвязными слоями, функции активации relu.
* CNN\_config\_2.py – вторая нейронная сеть с двумя сверточными слоями и с двумя полносвязными слоями, функции активации tanh.
* CNN\_config\_2.py – третья нейронная сеть с одним сверточным слоем и с двумя полносвязными слоями, функции активации tanh.
* CNN\_config\_4.py – четвертая нейронная сеть с одним сверточным слоем и с двумя полносвязными слоями, функция активации relu.

# Результаты экспериментов

В работе рассмотрены 4 конфигурации.

Параметры обучения:

количество эпох – 10,

скорость обучения – 0.001.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Конфигурация** | **Количество сверточных**  **слоев** | **Количество полносвязных слоев** | **Функции активации** | **Результат** | | |
| Точность на тренировочном множестве | Точность на тестовом множестве | Время, с |
| 1 | 2 | 2 | relu-relu-sigmoid | 0.94 | 0.94 | 127.69 |
| 2 | 2 | 2 | tanh-tanh-tanh-sigmoid | 0.74 | 0.75 | 127.93 |
| 3 | 1 | 2 | tanh-tanh-sigmoid | 0.63 | 0.69 | 166.73 |
| 4 | 1 | 2 | relu-relu-sigmoid | 0.99 | 0.98 | 167.03 |

# Анализ результатов

Указанный логарифм похож на ReLU.

Преимущества ReLU:

1. Вычисление сигмоиды и гиперболического тангенса требует ресурсоёмких операций, таких как возведение в степень, в то время как ReLU не подвержен насыщению.
2. 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 №4. 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.