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**

Направления подготовки: «Фундаментальная информатика и информационные технологии», «Прикладная математика и информатика»

Магистерские программы: «Компьютерная графика», «Вычислительные методы и суперкомпьютерные технологии»

Образовательный курс «Глубокое обучение»

**ОТЧЕТ**

по лабораторной работе №2

**Разработка полностью связанной нейронной сети**

**Выполнили:**

студенты группы 381706-3м

Бебнев Виктор

Голякова Елена

студенты группы 381703-3м

Митрохина Юлия

Береснева Юлия

Нижний Новгород

2018

Содержание

[Цели 4](#_Toc532478425)

[Задачи 4](#_Toc532478426)

[Постановка задачи 5](#_Toc532478427)

[Описание задачи 5](#_Toc532478428)

[Выбор библиотеки 5](#_Toc532478429)

[Тренировочные и тестовые наборы данных 6](#_Toc532478430)

[Метрики качества решения задачи 6](#_Toc532478431)

[Формат хранения данных 7](#_Toc532478432)

[Тестовые конфигурации сетей 8](#_Toc532478433)

[Разработанная программа 8](#_Toc532478434)

[Результаты и анализ экспериментов 9](#_Toc532478435)

[Эксперимент №1 9](#_Toc532478436)

[Эксперимент №2 11](#_Toc532478437)

[Эксперимент №3 13](#_Toc532478438)

[Выводы 15](#_Toc532478439)

# Цели

Цель: получить базовые навыки работы с одной из библиотек глубокого обучения (TensorFlow) на примере полностью связанных нейронных сетей.

# Задачи

Выполнение практической работы предполагает решение следующих задач:

1. Выбор библиотеки для выполнения практических работ курса.

2. Выбор практической задачи компьютерного зрения для выполнения практических работ.

3. Разработка программ/скриптов для подготовки тренировочных и тестовых данных в формате, который обрабатывается выбранной библиотекой.

4. Разработка нескольких архитектур полностью связанных нейронных сетей (варьируются количество слоев и виды функций активации на каждом слое) в формате, который принимается выбранной библиотекой.

5. Обучение разработанных глубоких моделей.

6. Тестирование обученных глубоких моделей.

7. Публикация разработанных программ/скриптов в репозиторий на GitHub.

8. Подготовка отчета, содержащего минимальный объем информации по каждому этапу выполнения работы.

# Постановка задачи

## Описание задачи

В качестве задачи компьютерного зрения была выбрана задача классификации рукописных изображений, представленная набором данных The Quick Draw dataset – коллекция рисунков пользователей игры [Quick, Draw!](https://quickdraw.withgoogle.com/), содержащая 345 категорий изображений, из которых для лабораторной работы были взяты следующие 5 категорий по 5 000 примеров:

* *spoon - ложка,*
* *paintbrush - кисть,*
* *smiley face - улыбающееся лицо,*
* *wheel - колесо,*
* *bush - куст.*



Рис.1 Примеры рисунков из набора данных Quick, Draw!

Рисунки представлены набором точек – *(x, y, t)* - вектор координат пикселей на плоскости со значением времени первой точки и разметкой, включающей в себя информацию о стране игрока и о том, что требовалось нарисовать.

## Выбор библиотеки

Для выполнения данной практической работы была выбрана библиотека глубокого обучения TensorFlow, использующая в качестве интерфейса язык программирования Python.

Для проверки корректности установки библиотеки была выполнена разработка и запуск тестового примера сети для задачи классификации рукописных цифр из набора данных MNIST. На данном примере достигнутая точность равна 0,931.

## Тренировочные и тестовые наборы данных

В используемом наборе данных 25 000 (5 категорий по 5 000 примеров) изображений, из которых использовали как:

* тренировочных – 17 500 (70%);
* валидационных – 2 500 (10%);
* тестовых – 5 000 (20%)

В ходе обучения наборы тщательно перемешиваются перед выборками.

## Метрики качества решения задачи

Качество решения выбранной задачи оценивается с использованием различных метрик:

* accuracy – точность – это отношение числа верно классифицированных изображений к общему числу изображений в выборке

Введем некоторые обозначения для определения следующих величин:

* TP — истино-положительное решение;
* TN — истино-отрицательное решение;
* FP — ложно-положительное решение;
* FN — ложно-отрицательное решение;

тогда можем определить вычисление следующих метрик по формулам:

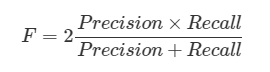
* precision – точность – это доля изображений действительно принадлежащих данному классу относительно количества всех изображений, которые сеть отнесла к этому классу



* recall – полнота – это доля найденных сетью изображений, принадлежащих классу относительно количества всех изображений этого класса в тестовой выборке



* f1-score – F-мера – это гармоническое среднее между точностью и полнотой



Формула этой метрики позволяет одновременно учитывать значения точности и полноты, поскольку придает им одинаковый вес, а значит будет одинаково падать при изменении любой из них.

* support – это

Известно, что чем выше показатели точности и полноты, тем лучше классификатор.

## Формат хранения данных

Исходный формат хранения данных - используемый набор данных хранится на сервере-источнике в виде бинарных файлов:

* <https://console.cloud.google.com/storage/browser/quickdraw_dataset/full/binary>
* <https://drive.google.com/drive/folders/1ku5w3lUnQnr8cmA4J9Ra9yV5Xd7y7awk>

в формате, указанном в Таблице 1 ниже:

**Таблица 1**

**Исходный формат хранения данных**

|  |  |  |
| --- | --- | --- |
| **Атрибут** | **Тип** | **Описание** |
| key\_id | 64-bit unsigned integer | Уникальный идентификатор |
| word | string | Категория изображения |
| recognized | boolean | Указано, что слово было опознано в игре |
| timestamp | datetime | Дата создания рисунка |
| countrycode | string | Код страны участника |
| drawing | string | JSON массив, содержащий вектор рисования |

Формат данных на входе сети – в ходе проведения экспериментов данные подавались в следующих видах:

* 3-мерный массив нормализованных координат данных, длины равной максимальной длине набора точек рисунка (недостающие элементы заполнялись нулями);
* нормализованные бинарные изображения, отмасштабированные по размеру 28х28, полученные с помощью библиотеки OpenCV из данных источника;
* в ходе подготовки данных все данные разметки были векторизованы.

# Разработанная программа

Разработанная программа содержит следующие файлы в директории src:

* parse\_data.py - подготовка данных;
* run\_create\_dataset.py - выбор данных, которые распознала сеть Google, распределение на run. train и validate выборки;
* run\_train\_conv.py - непосредственно обучение сети, использует класс NetworkBase;
* NetworkBase.py - описание нейронной сети;
* run\_statisctic\_conv.py - получение статистики по итогам обучения.

# Тестовые конфигурации сетей

В ходе лабораторной работы нами была разработана архитектура свёрточной нейронной сети в формате, принимаемом библиотекой глубокого обучения TensorFlow. Ее структура имеет следующий вид:

*n* слоев

*S* входных нейронов

*l* выходных нейронов

*k* скрытых слоев

Рис.2 Схема тестовой конфигурации сети

В качестве функции активации использовалась функция ReLU:

f(x) = min(max(0, x),6).

В качестве функции потерь использовалась функция cross entropy.

H(p,q) = - \sum_x p(x) \log(q(x))

# Результаты и анализ экспериментов

В ходе работы были проведены 3 эксперимента на машине со следующими характеристиками:

* операционная система Ubuntu 16.04, CUDA 9.1
* оперативная память RAM 60 GB
* видеокарта Geforce GTX 1060 6 GB x2 Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz.

Приведем проведенные эксперименты:

## Эксперимент №1

Рассматривалась сеть из 1 слоя, функция активации ReLU6, функция потерь Cross entropy, количество эпох 50, скорость обучения 0.001 оптимизатор *GradientDescentOptimizer*.

График функции ошибки:

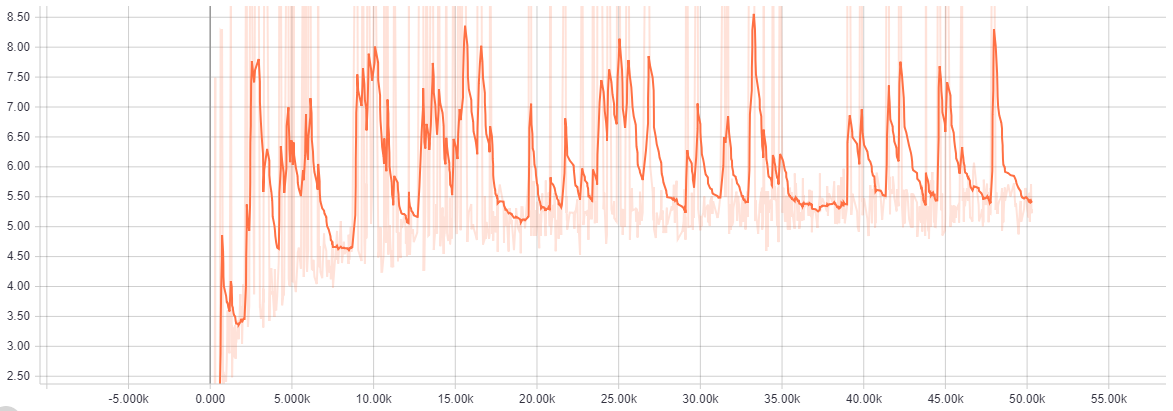


Рис.3 График функции ошибки в эксперименте №1

График точности:

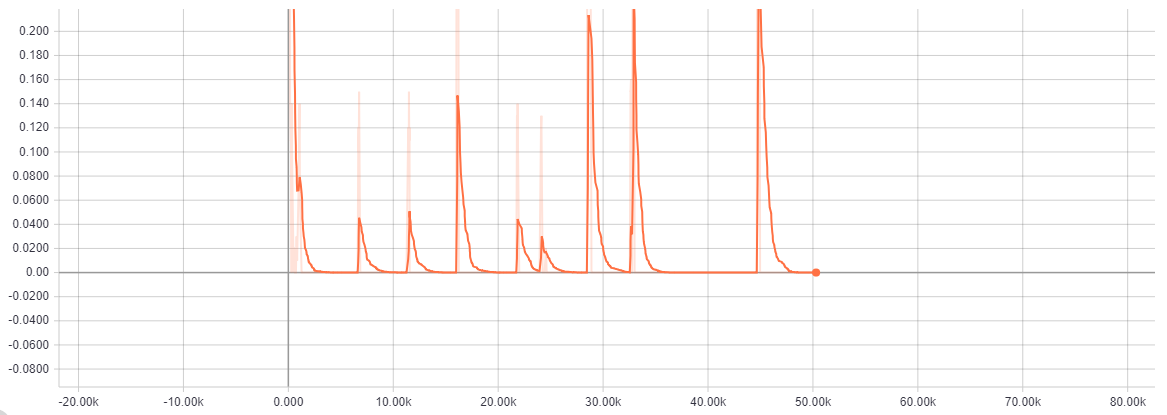


Рис.4 График точности в эксперименте №1

**Таблица 2**

**Значения метрик в эксперименте №1**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **precision** | **recall** | **f1-score** | **support** |
| **spoon** | 0.87 | 0.94 | 0.90 | 925 |
| **paintbrush** | 0.96 | 0.83 | 0.89 | 1150 |
| **smiley face** | 0.93 | 0.88 | 0.90 | 1057 |
| **wheel** | 0.97 | 0.89 | 0.93 | 1089 |
| **bush** | 0.73 | 0.94 | 0.82 | 779 |
| **micro avg** | 0.89 | 0.89 | 0.89 | 5000 |
| **macro avg** | 0.89 | 0.90 | 0.89 | 5000 |
| **weighted avg** | 0.90 | 0.89 | 0.89 | 5000 |

## Эксперимент №2

Рассматривалась сеть из 4 слоев, функция активации ReLU6, функция потерь Cross entropy, количество эпох 50, скорость обучения 0.001 оптимизатор *GradientDescentOptimizer*. Первые 13 эпох batch size = 8, далее и до конца - 4.

График функции ошибки:

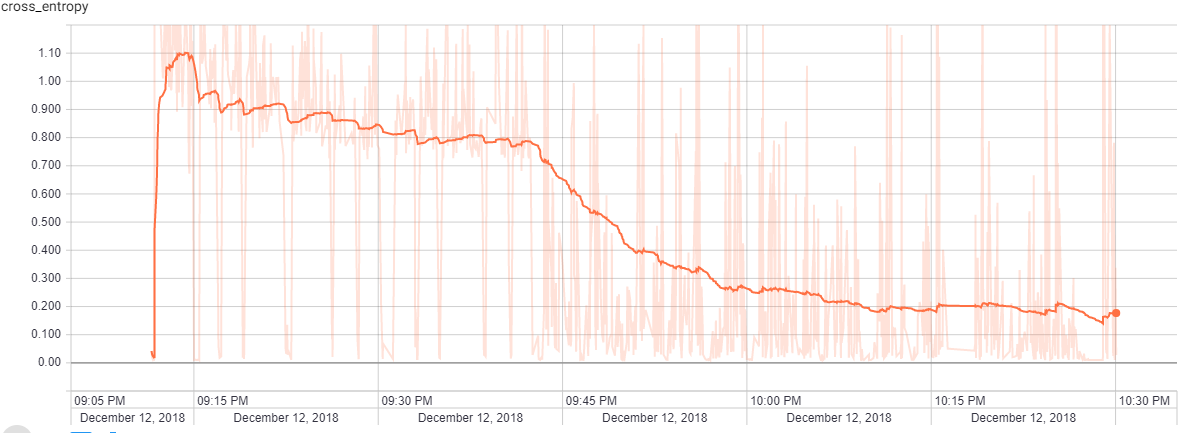


Рис.5 График функции ошибки в эксперименте №2

График точности:

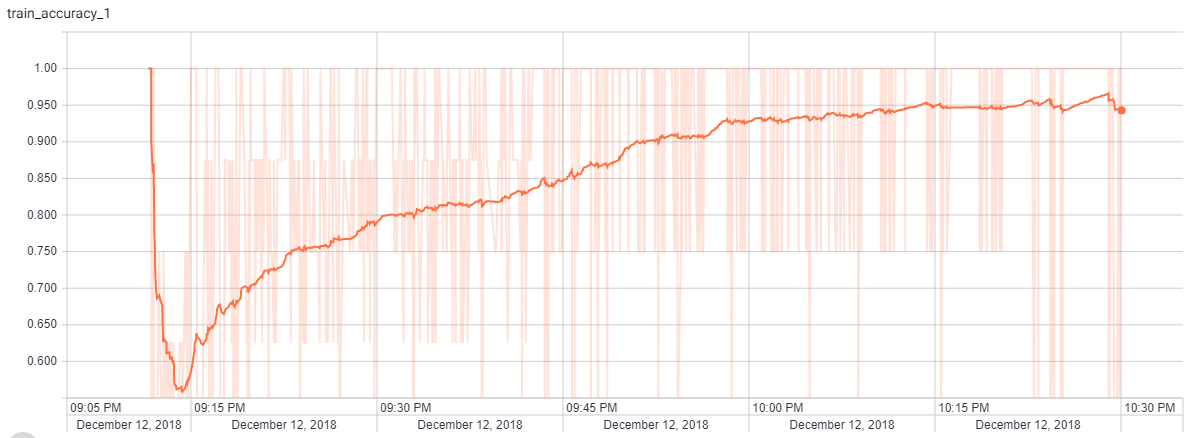


Рис.6 График точности в эксперименте №2

**Таблица 3**

**Значения метрик в эксперименте №2**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **precision** | **recall** | **f1-score** | **support** |
| **spoon** | 0.93 | 0.94 | 0.93 | 983 |
| **paintbrush** | 0.94 | 0.90 | 0.92 | 1040 |
| **smiley face** | 0.97 | 0.76 | 0.75 | 1282 |
| **wheel** | 0.97 | 0.93 | 0.95 | 1039 |
| **bush** | 0.64 | 0.97 | 0.77 | 656 |
| **micro avg** | 0.89 | 0.89 | 0.89 | 5000 |
| **macro avg** | 0.89 | 0.90 | 0.89 | 5000 |
| **weighted avg** | 0.91 | 0.89 | 0.89 | 5000 |

## Эксперимент №3

Рассматривалась сеть из 3 слоев, функция активации ReLU6, функция потерь Cross entropy, количество эпох 50, скорость обучения 0.001, оптимизатор *GradientDescentOptimizer*.

График функции ошибки:

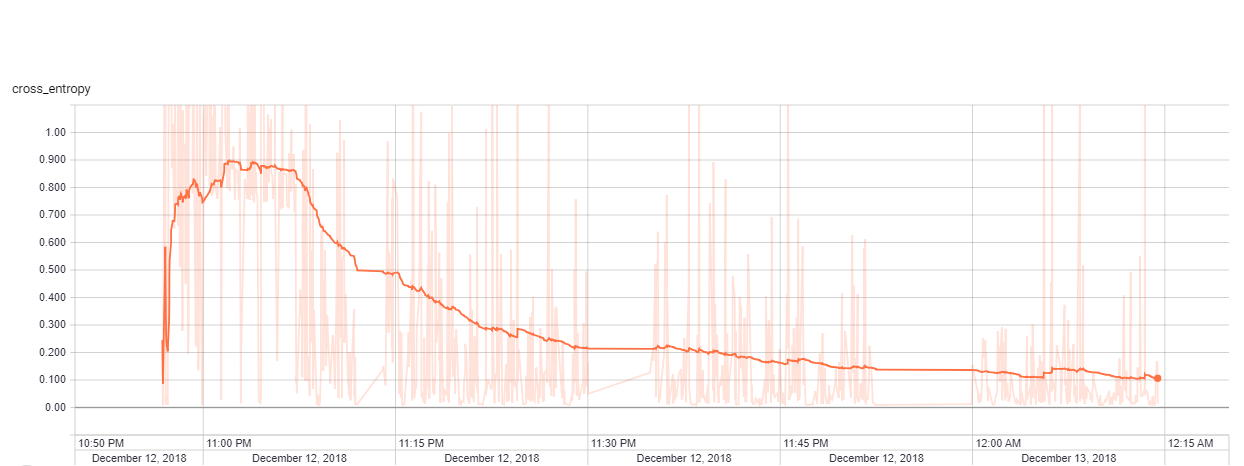


Рис.7 График функции ошибки в эксперименте №3

График точности:

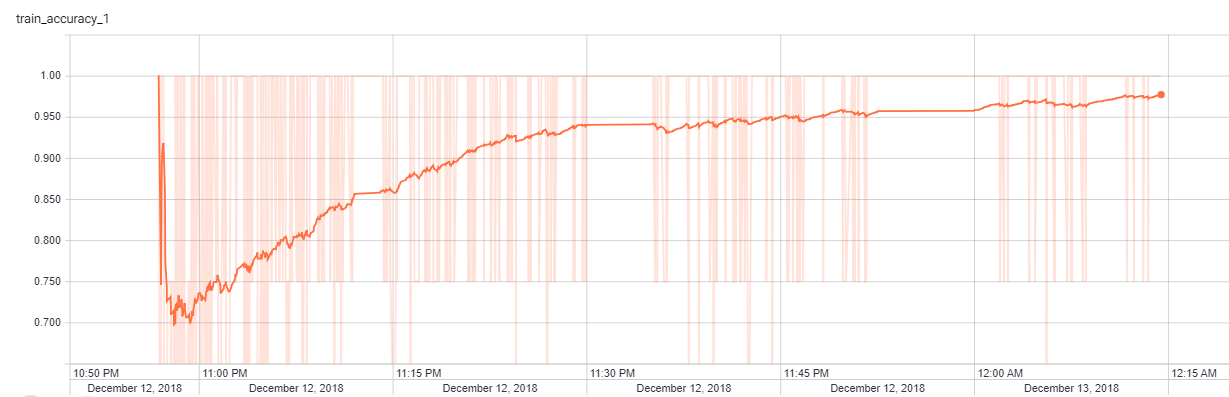


Рис.8 График точности в эксперименте №3

**Таблица 4**

**Значения метрик в эксперименте №3**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **precision** | **recall** | **f1-score** | **support** |
| **spoon** | 0.97 | 0.94 | 0.95 | 1034 |
| **paintbrush** | 0.95 | 0.92 | 0.94 | 1034 |
| **smiley face** | 0.97 | 0.87 | 0.92 | 1107 |
| **wheel** | 0.98 | 0.94 | 0.96 | 1038 |
| **bush** | 0.75 | 0.96 | 0.84 | 787 |
| **micro avg** | 0.92 | 0.92 | 0.92 | 5000 |
| **macro avg** | 0.92 | 0.93 | 0.92 | 5000 |
| **weighted avg** | 0.93 | 0.92 | 0.93 | 5000 |

**Таблица 5**

**Сводная таблица результатов обучения по проведенным экспериментам**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Число скрытых нейронов, послойно** | **Результаты обучения** | | |
| Точность | Ошибка | Время |
| **1** | [10350,  10350,  5175,  2070,  1035,  690] | - | - | 2ч 48 мин |
| **2** | [3\*28\*28,  2\*28\*28,  14\*28] | 0.96 | 0.14 | 1ч 13 мин |
| **3** | [2\*28\*28,  14\*28] | 0.98 | 0.1 | 1ч 17 мин |

# Выводы

В ходе лабораторной работы получили базовые навыки работы с библиотекой глубокого обучения TensorFlow на примере решения задачи [Quick, Draw!](https://quickdraw.withgoogle.com/). С помощью выбранной библиотеки построили различные конфигурации полностью связанных нейронных сетей для решения практических задач.

На основе построенных архитектур разработали программы для обучения глубоких моделей. Затем было проведено обучение модели и тестирование на наборе данных задачи.

Собранные метрики, показатели и результаты были визуализированы с помощью модуля TensorBoard библиотеки TensorFlow и представлены в виде графиков, приведенных выше в данном отчёте.

Был проведен анализ полученных результатов, из которого следует вывод: наилучший результат со значением точности 0.98 был достигнут на конфигурации полносвязной сети, описанной в эксперименте №3.

Таким образом, цель работы - получить базовые навыки работы с библиотекой глубокого обучения (TensorFlow) на примере полностью связанных нейронных сетей - была достигнута.