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 №3

**Разработка сверточных нейронных сетей**

**Выполнили:**

студенты группы 381706-3м

Бебнев Виктор

Голякова Елена

студенты группы 381703-3м

Митрохина Юлия

Береснева Юлия

Нижний Новгород

2018

Содержание

[Цели 3](#_Toc532554156)

[Задачи 3](#_Toc532554157)

[Постановка задачи 4](#_Toc532554158)

[Описание задачи 4](#_Toc532554159)

[Выбор библиотеки 4](#_Toc532554160)

[Тренировочные и тестовые наборы данных 5](#_Toc532554161)

[Метрики качества решения задачи 5](#_Toc532554162)

[Формат хранения данных 7](#_Toc532554163)

[Разработанная программа 8](#_Toc532554164)

[Тестовые конфигурации сетей 8](#_Toc532554165)

[Конфигурация №1 9](#_Toc532554166)

[Конфигурация №2 10](#_Toc532554167)

[Конфигурация №3 11](#_Toc532554168)

[Результаты и анализ экспериментов 12](#_Toc532554169)

[Эксперимент №1 12](#_Toc532554170)

[Эксперимент №2 14](#_Toc532554171)

[Эксперимент №3 16](#_Toc532554172)

[Выводы 19](#_Toc532554173)

# Цели

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

# Задачи

Выполнение практической работы предполагает решение следующих задач:

1. Разработка нескольких архитектур сверточных нейронных сетей (варьируются количество слоев и виды функций активации на каждом слое) в формате, который принимается библиотекой глубокого обучения TensorFlow.
2. Обучение разработанных глубоких моделей.
3. Тестирование обученных глубоких моделей.
4. Публикация разработанных программ/скриптов в репозитории на GitHub.
5. 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 –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. Отметим, что чем выше показатели точности и полноты, тем лучше классификатор.

* support – это количество примеров, классифицированных к определенному классу в выборке;

Таким образом, по классам в сумме значение этой метрики равняется числу изображений в выборке, конкретно в работе 5000 изображений.

При наличии нескольких наборов данных и высчитанных для них значений метрик precision и recall, можно высчитать также метрики:

* micro avg – это micro средние значения, рассчитываемые для precision и recall:

где *c* – отметка класса.

Из формул следует, что при нескольких классах значение значения micro-метрик всегда совпадают, поскольку .

* macro avg – это macro средние значения, рассчитываемые для precision и recall:

где *c* – отметка класса, *#c* – число классов.

* weighed avg – это взвешенные средние значения, рассчитываемые для precision и recall с весами:

где - количество примеров в классе.

## 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 основных методов для работы с нейронной сетью;
* ConvNetwork.py – описание архитектуры полносвязной нейронной сети;
* run\_statisctic\_conv.py - получение статистики по итогам обучения.

# Тестовые конфигурации сетей

В ходе лабораторной работы нами были разработаны 3 архитектуры свёрточных нейронных сетей в формате, принимаемом библиотекой глубокого обучения TensorFlow. Их структуру можно представить в следующем виде:

|  |  |
| --- | --- |
| Рис.2 Схема тестовых архитектур сетей | Последний полносвязный слой в каждой архитектуре состоит из 5 нейронов. Описание каждой конфигурации представлено далее. Ядра сверточных слоев 3x3, шаг = 1. Размер pooling равен 2x2, смещается с шагом = 2.  В экспериментах использовались следующие функции:   * функция активации ReLU6: * операция пространственного объединения – *pooling*; * функция потерь - cross entropy * функция активации на полносвязном слое – *softmax*. |

## Конфигурация №1

|  |  |
| --- | --- |
| Screenshot_4_1.png  Рис.3 Конфигурация №1 | Рассматривалась сеть из 3-х блоков:   * каждый блок состоит из   + 2 свёрточных слоёв,   + завершается max pooling’ом * количество фильтров у свёрточных слоёв   + в первых двух блоках - 16,   + в третьем - 32. * ядра сверточных слоев – 3х3, шаг = 1 * операция пространственного объединения pooling - 2x2, шаг = 2 * последний - полносвязный слой состоит из 5 нейронов   Использованы в ходе:   * функция активации - ReLU6 * функция потерь - cross entropy * функция активации на полносвязном слое – softmax |

## Конфигурация №2

|  |  |
| --- | --- |
| Screenshot_5_1.png  Рис.4 Конфигурация №2 | Рассматривалась сеть из 4-х блоков:   * каждый блок состоит из   + 2 свёрточных слоёв,   + завершается max pooling’ом * количество фильтров у свёрточных слоёв   + в первых двух блоках - 16,   + в последних двух - 32. * ядра сверточных слоев – 3х3, шаг = 1 * операция пространственного объединения pooling - 2x2, шаг = 2 * последний - полносвязный слой состоит из 5 нейронов   Использованы в ходе:   * функция активации - ReLU6 * функция потерь - cross entropy * функция активации на полносвязном слое – softmax |

## Конфигурация №3

|  |  |
| --- | --- |
| Screenshot_6_1.png  Рис.5 Конфигурация №3 | Рассматривалась сеть из 3-х блоков:   * каждый блок состоит из   + 2 свёрточных слоёв,   + завершается max pooling’ом * количество фильтров у свёрточных слоёв   + в первых двух блоках - 16,   + в последних двух - 32. * ядра сверточных слоев – 3х3, шаг = 1 * слой нормализации пачки – batch normalization * операция пространственного объединения pooling - 2x2, шаг = 2 * последний - полносвязный слой состоит из 5 нейронов   Использованы в ходе:   * функция активации - ReLU6 * функция потерь - cross entropy * функция активации на полносвязном слое – softmax |

# 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.
* Python 3.6, TensorFlow 1.12;

Приведем проведенные эксперименты:

## Эксперимент №1

С помощью разработанной программы в ходе эксперимента с конфигурацией №1 были собраны, визуализированы и проанализированы метрики качества решения для описанной задачи.

На рис. 6 представлен график изменения значения функции потерь cross entropy в течение эксперимента.

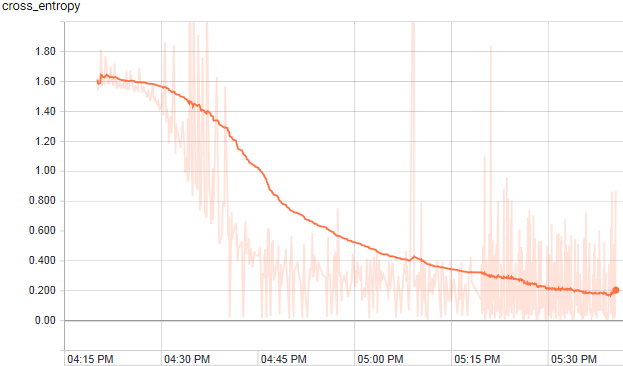


Рис.6 График функции ошибки в эксперименте №1

График показывает уменьшение значения функции потерь с течением времени и достижения отметки, равной 0.2, что является достаточно неплохим результатом.

На рис. 7 представлен график изменения значения точности (accuracy) в течение эксперимента. Из данных графика виден скачок вниз по оси ординат, затем с течением времени точность возрастает с неравномерной скоростью и достигает значения, равного 0.95.

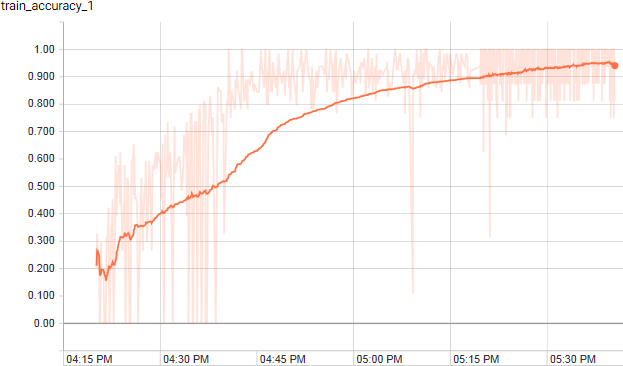


Рис.7 График точности в эксперименте №1

В Таблице 2 представлены значения метрик качества решения задачи в эксперименте №1 по каждому из представленных классов изображений решаемой задачи, а также их средние значения.

**Таблица 2**

**Значения метрик в эксперименте №1**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **precision** | **recall** | **f1-score** | **support** |
| **spoon** | 0.94 | 0.93 | 0.93 | 1013 |
| **paintbrush** | 0.95 | 0.85 | 0.90 | 1123 |
| **smiley face** | 0.94 | 0.93 | 0.93 | 1014 |
| **wheel** | 0.98 | 0.93 | 0.95 | 1055 |
| **bush** | 0.77 | 0.97 | 0.86 | 795 |
| **micro avg** | 0.92 | 0.92 | 0.92 | 5000 |
| **macro avg** | 0.92 | 0.92 | 0.25 | 5000 |
| **weighted avg** | 0.92 | 0.92 | 0.92 | 5000 |

Значения статистических метрик достаточно высокие, почти каждый из показателей переваливает отметку в 0.90. Значения **support** показывают, что в выборке в достаточном количестве представлены примеры каждого класса.

## Эксперимент №2

С помощью разработанной программы в ходе эксперимента с конфигурацией №2 были собраны, визуализированы и проанализированы метрики качества решения для описанной задачи.

На рис. 8 представлен график изменения значения функции потерь cross entropy в течение эксперимента.

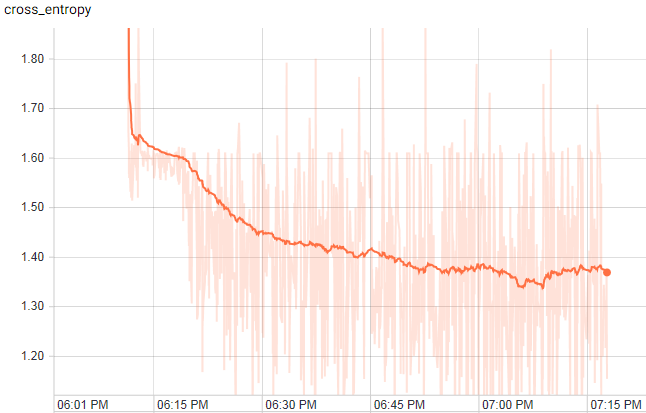


Рис.8 График функции ошибки в эксперименте №2

График показывает слабое уменьшение значения функции потерь с течением времени и достижения отметки, равной 1.37, что является достаточно плохим результатом.

На рис. 9 представлен график изменения значения точности (accuracy) в течение эксперимента.

По графику видно, что значение точности скачкообразно увеличивается с течением времени и достигает всего лишь значения, равного 0.35.

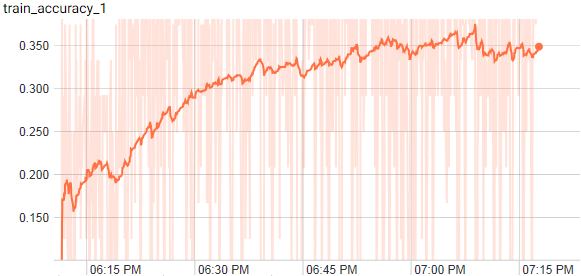


Рис.9 График точности в эксперименте №2

В Таблице 3 представлены значения метрик качества решения задачи в эксперименте №2 по каждому из представленных классов изображений решаемой задачи, а также их средние значения.

**Таблица 3**

**Значения метрик в эксперименте №2**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **precision** | **recall** | **f1-score** | **support** |
| **spoon** | 0.93 | 0.24 | 0.39 | 3806 |
| **paintbrush** | 0.94 | 0.79 | 0.86 | 1194 |
| **smiley face** | 0.00 | 0.00 | 0.00 | 0 |
| **wheel** | 0.00 | 0.00 | 0.00 | 0 |
| **bush** | 0.00 | 0.00 | 0.00 | 0 |
| **micro avg** | 0.37 | 0.37 | 0.37 | 5000 |
| **macro avg** | 0.37 | 0.21 | 0.25 | 5000 |
| **weighted avg** | 0.93 | 0.37 | 0.50 | 5000 |

Исходя из значений метрик в таблице, можно сказать, что показатели качества достаточно низкие, что говорит также о низком качестве классификатора.

## Эксперимент №3

С помощью разработанной программы в ходе эксперимента с конфигурацией №3 были собраны, визуализированы и проанализированы метрики качества решения для описанной задачи.

На рис. 10 представлен график изменения значения функции потерь cross entropy в течение эксперимента.

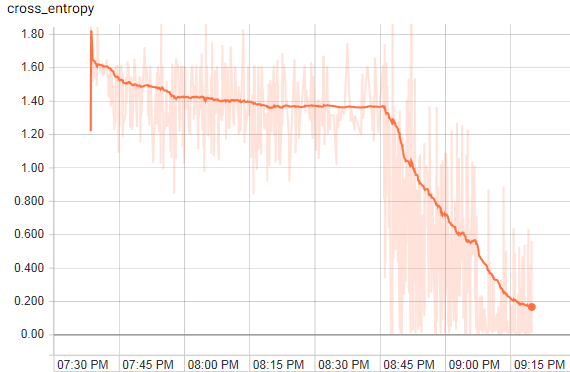


Рис.10 График функции ошибки в эксперименте №3

График сначала показывает плавное уменьшение значения функции потерь с течением времени, затем резкое уменьшение и достижения отметки, равной 0,18, что является достаточно хорошим результатом. Момент изменения скорости уменьшения значения на графике соответствует моменту изменения размера пакет, параметр batch\_size сменился со значения 16 до 8.

На рис. 11 представлен график изменения значения точности (accuracy) в течение эксперимента.

По графику видно, что значение точности медленно и скачкообразно увеличивается при batch\_size=16, а затем быстро растет установления batch\_size=8 и с течением времени достигает значения, равного 0.95.

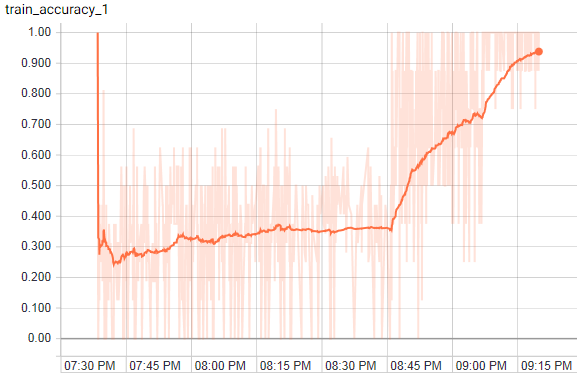


Рис.11 График точности в эксперименте №3

В Таблице 4 представлены значения метрик качества решения задачи в эксперименте №3 по каждому из представленных классов изображений решаемой задачи, а также их средние значения.

**Таблица 4**

**Значения метрик в эксперименте №3**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **precision** | **recall** | **f1-score** | **support** |
| **spoon** | 0.98 | 0.94 | 0.96 | 1044 |
| **paintbrush** | 0.95 | 0.94 | 0.95 | 1015 |
| **smiley face** | 0.98 | 0.95 | 0.96 | 1024 |
| **wheel** | 0.97 | 0.97 | 0.97 | 1000 |
| **bush** | 0.88 | 0.96 | 0.92 | 917 |
| **micro avg** | 0.95 | 0.95 | 0.95 | 5000 |
| **macro avg** | 0.95 | 0.95 | 0.95 | 5000 |
| **weighted avg** | 0.95 | 0.95 | 0.95 | 5000 |

Из таблицы видно, что значения метрик достаточно хорошие, почти каждый из показателей переваливает отметку в 0.92, что говорит о высоком качестве построенного классификатора.

Отметим, что особенностью конфигурации №3 является наличие слоя нормализации пакета на последнем этапе каждого блока. Считаем, что это достигается за счет известного факта, что batch normalization помогает ускорить процесс обучения, поворачивая активацию на единицу Гауссовского распределения и тем самым решая проблему потери градиентов.

Проведем сравнительный анализ полученных результатов в ходе проведенных экспериментов и опишем его в виде Таблицы 5.

**Таблица 5**

**Сводная таблица результатов обучения по проведенным экспериментам**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Число блоков** | **Результаты обучения** | | |
| Точность | Значение функции ошибки (cross entropy) | Время |
| **1** | 3 | 0.95 | 0.2 | 1ч 15 мин |
| **2** | 4 | 0.35 | 1.37 | 1ч 05 мин |
| **3** | 3 | 0.95 | 0.18 | 1ч 45 мин |

Таблица 5 позывает, что слой нормализации дает пусть и небольшой, но выигрыш в функции потерь и достаточно хорошее значение точности. Возможно, при масштабируемости данная конфигурация покажет еще лучшие показатели.

# Выводы

В ходе лабораторной работы построили несколько архитектур свёрточных нейронных сетей, которые позволяют решать практическую задачу с высокими показателями качества.

На основе построенных архитектур разработали программы для обучения глубоких моделей. Затем было проведено обучение модели и тестирование на наборе данных задачи [Quick, Draw!](https://quickdraw.withgoogle.com/).

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

В ходе экспериментов было установлено, что к наилучшим результатам привела сеть с конфигурацией №3, содержащая слои нормализации пакетов – batch normalization.

Таким образом, цель работы - построить архитектуру свёрточной нейронной сети, которая позволяет решать практическую задачу с высокими показателями качества - была достигнута.