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](#_Toc532479356)

[Задачи 3](#_Toc532479357)

[Постановка задачи 4](#_Toc532479358)

[Описание задачи 4](#_Toc532479359)

[Выбор библиотеки 4](#_Toc532479360)

[Тренировочные и тестовые наборы данных 5](#_Toc532479361)

[Метрика качества решения задачи 5](#_Toc532479362)

[Формат хранения данных 6](#_Toc532479363)

[Тестовые конфигурации сетей 7](#_Toc532479364)

[Разработанная программа 8](#_Toc532479365)

[Результаты и анализ экспериментов 9](#_Toc532479366)

[Эксперимент №1 9](#_Toc532479367)

[Эксперимент №2 11](#_Toc532479368)

[Эксперимент №3 12](#_Toc532479369)

[Выводы 14](#_Toc532479370)

# 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, которые используются в качестве:

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

В ходе обучения наборы тщательно перемешиваются перед выборками.

## Метрика 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:

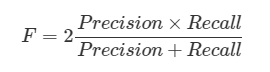
* precision – точность – это доля изображений действительно принадлежащих данному классу относительно количества всех изображений, которые сеть отнесла к этому классу



* 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 количество примеров каждого класса в размеченных данных.

## 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 были разработаны 3 архитектуры свёрточных нейронных сетей в формате, принимаемом библиотекой глубокого обучения TensorFlow.

Их структуру можно представить в следующем виде:

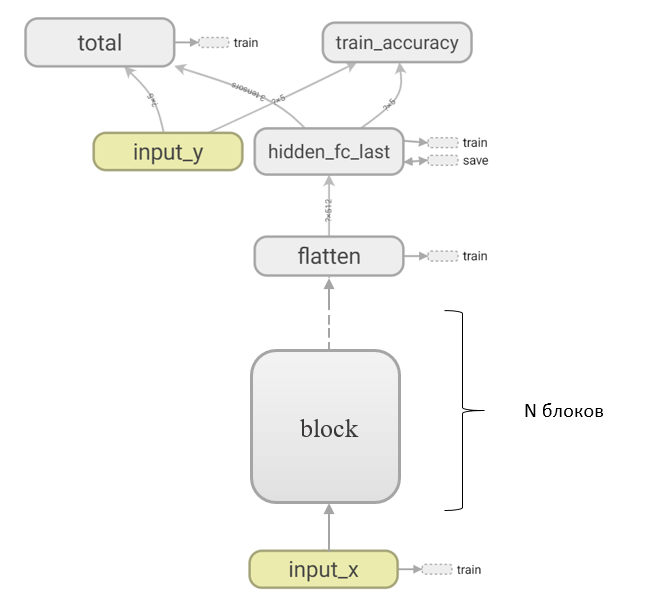


Рис.2 Схема тестовых архитектур сетей

Последний полносвязный слой состоит из 5 нейронов. Описание блоков будет представлено в разделе "Результаты и анализ экспериментов".

Ядра сверточных слоев имеют размер 3x3, смещаются с шагом 1.

Размер pooling равен 2x2, смещается с шагом 2.

В экспериментах использовались следующие функции:

* функция активации ReLU6

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

* операция пространственного объединения – pooling;
* функция потерь - cross entropy

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

* 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.

Приведем проведенные эксперименты:

## Эксперимент №1

Рассматривалась сеть из 3-х блоков, каждый из которых состоит из 2 свёрточных слоёв и завершается max pooling. Количество фильтров у свёрточных слоёв в первых двух блоках - 16, в третьем - 32.

Значение функции ошибки стремится к 0.2:

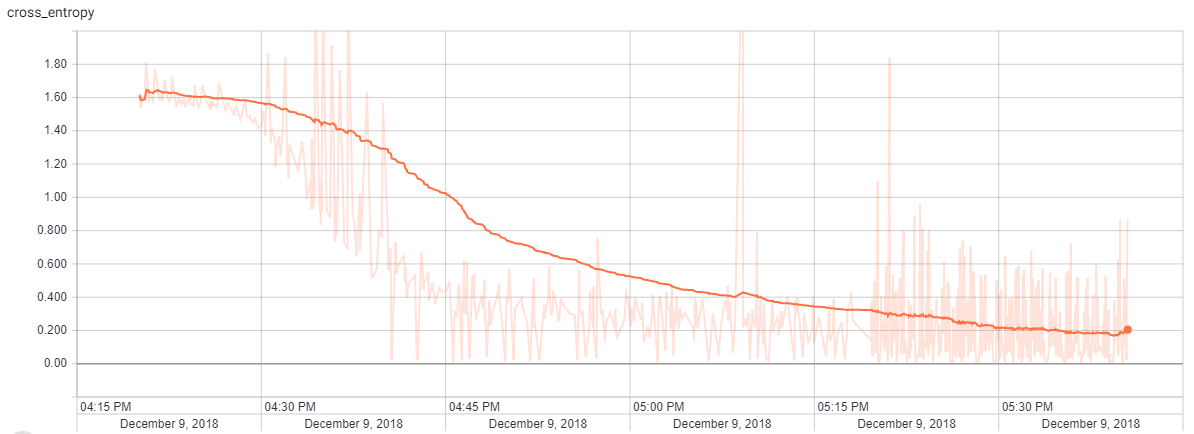


Рис.3 График функции ошибки в эксперименте №1

Точность стремится к 0.95:

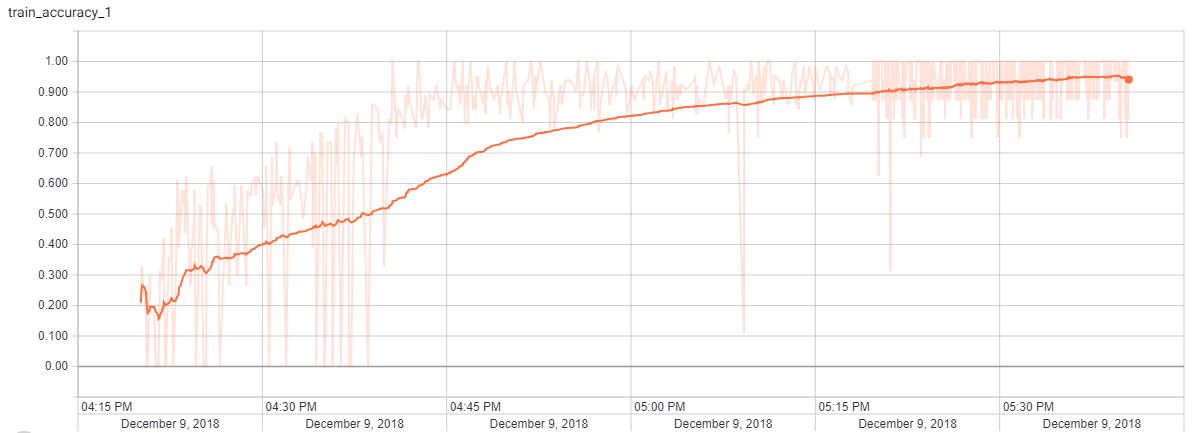


Рис.4 График точности в эксперименте №1

**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 |

## Эксперимент №2

Рассматривалась сеть из 4-х блоков, каждый из которых состоит из 2 свёрточных слоёв и завершается max pooling. Количество фильтров у свёрточных слоёв в первых двух блоках - 16, в третьем и четвертом - 32.

Значение функции ошибки стремится к 1.37:

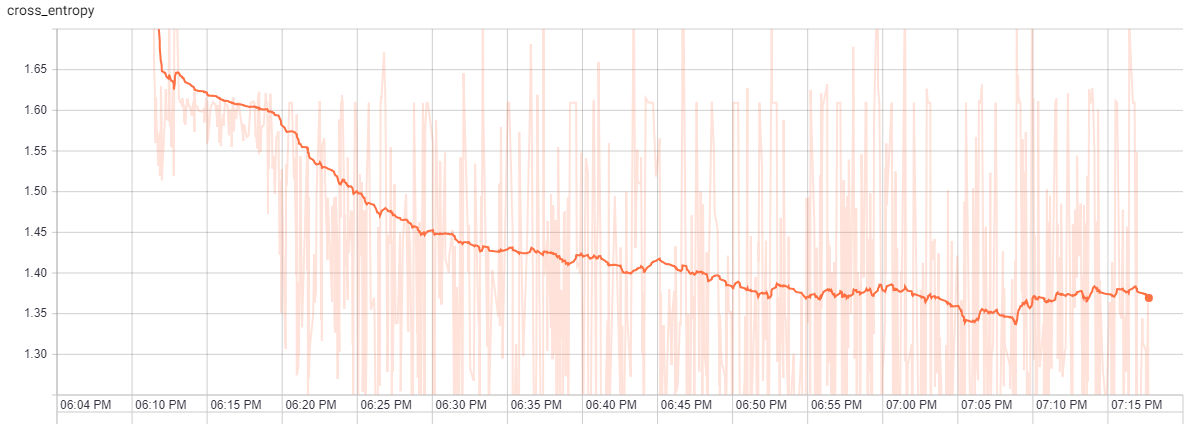


Рис.5 График функции ошибки в эксперименте №2

Точность стремится к 0.35:

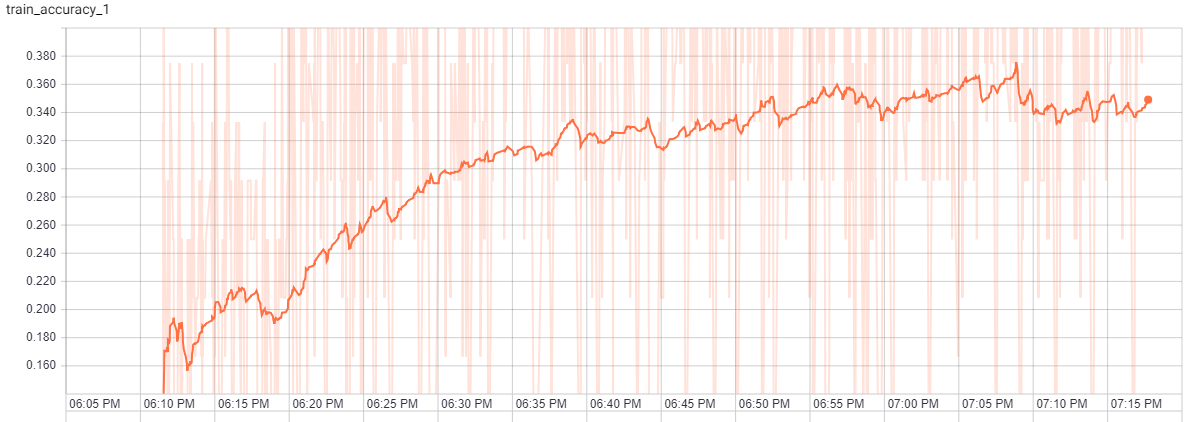


Рис.6 График точности в эксперименте №2

**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 |

## Эксперимент №3

Рассматривалась сеть из 3-х блоков, каждый из которых состоит из 2 свёрточных слоёв, добавлен batch normalization, завершается блок max pooling.

Batch Normalization помогает ускорить процесс обучения, поворачивая активацию на единицу Гауссовского распределения и тем самым решая проблему потери градиентов.

Количество фильтров в первых двух блоках - 16, в третьем - 32.

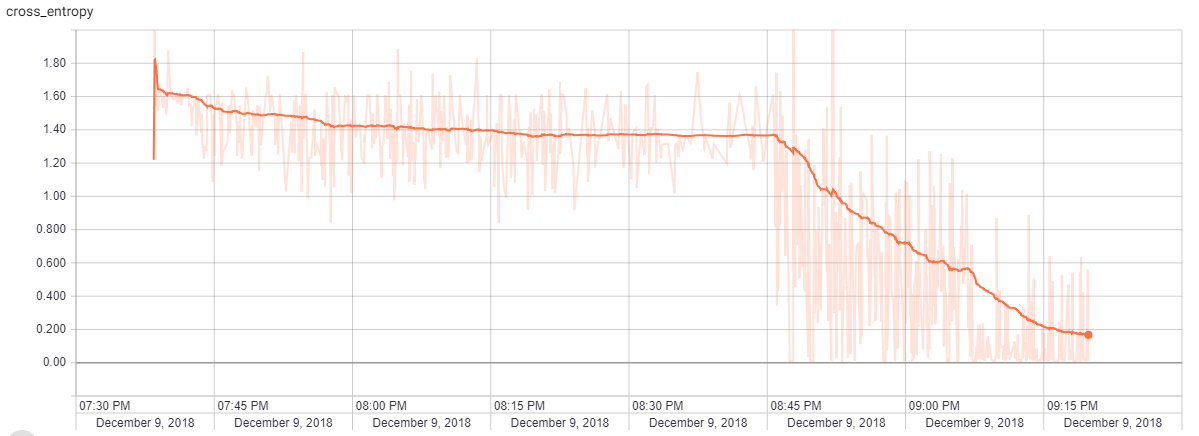
Значение функции ошибки стремится к 0.18:

Рис.7 График функции ошибки в эксперименте №3

Точность стремится к 0.95:

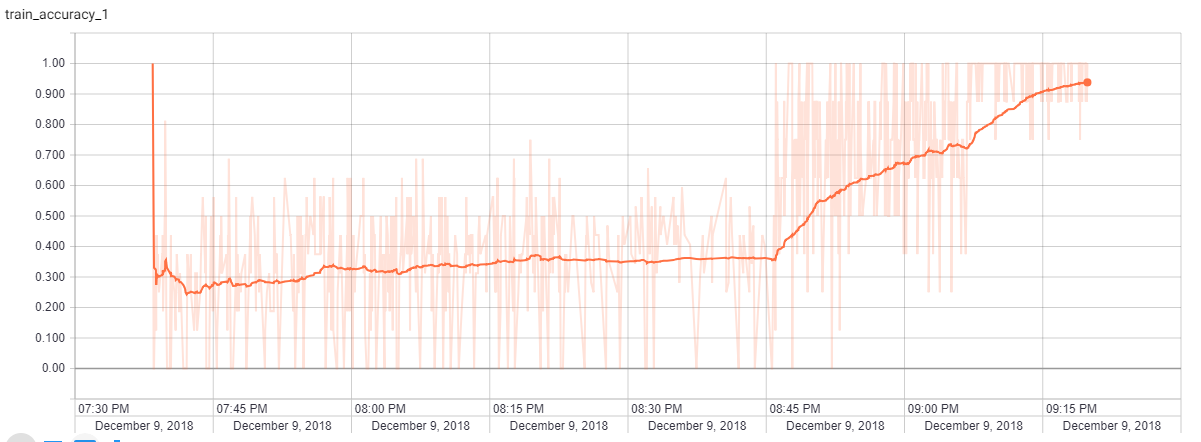


Рис.8 График точности в эксперименте №3

На графиках можно увидеть резкое падение значения функции ошибки и рост точности, это связано с уменьшением batch size c 16 до 8.

**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 |

**Таблица 5**

**Сводная таблица результатов обучения по проведенным экспериментам**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Число блоков** | **Число сверточных слоев в блоке** | **Наличие batch normalization** | **Результаты обучения** | | |
| Точность | Ошибка | Время |
| **1** | 3 | 2 | **-** | 0.95 | 0.2 | 1ч 15 мин |
| **2** | 4 | 2 | **-** | 0.35 | 0.63 | 1ч 05 мин |
| **3** | 3 | 3 | **+** | 0.95 | 0.18 | 1ч 45 мин |

# 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 под номером 3, также описанная выше.

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