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 №5

**Применение переноса обучения для решения задачи классификации изображений игры Quick, Draw! с помощью библиотеки TensorFlow**

**Выполнили:**

студенты группы 381706-3м

Бебнев Виктор

Голякова Елена

студенты группы 381703-3м

Митрохина Юлия

Береснева Юлия

Нижний Новгород

2018

Содержание

[Цели 3](#_Toc533074136)

[Задачи 3](#_Toc533074137)

[Постановка задачи 4](#_Toc533074138)

[Описание задачи 4](#_Toc533074139)

[Выбор библиотеки 4](#_Toc533074140)

[Тренировочные и тестовые наборы данных 5](#_Toc533074141)

[Метрики качества решения задачи 5](#_Toc533074142)

[Формат хранения данных 7](#_Toc533074143)

[Разработанная программа 8](#_Toc533074144)

[Тестовые конфигурации сетей 8](#_Toc533074145)

[Конфигурация №1 Прямое использование обученной модели 8](#_Toc533074146)

[Конфигурация №2 Использование структуры обученной модели 9](#_Toc533074147)

[Конфигурация №3 Фиксированный метод извлечения признаков модели 10](#_Toc533074148)

[Результаты и анализ экспериментов 12](#_Toc533074149)

[Выводы 13](#_Toc533074150)

# Цели

Цель: исследовать возможности переноса обучения для решения целевой задачи, выбранной изначально для выполнения работ.

# Задачи

Выполнение практической работы предполагает решение следующих задач:

1. Поиск исходной задачи, близкой по смыслу к целевой задаче.
2. Поиск натренированной модели для решения исходной задачи.
3. Выполнение трех типов экспериментов по переносу знаний, описанных в лекциях.
4. Сбор результатов экспериментов.

# 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.

## 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 227х227x3, где 3 – количество каналов, 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\_transfer.py - непосредственно обучение сети, использует класс NetworkBase;
* NetworkBase.py - описание основных методов для работы с нейронной сетью;
* TransferNetwork.py – описание архитектуры сверточной нейронной сети;
* run\_statisctic\_transfer.py - получение статистики по итогам обучения.

# Тестовые конфигурации сетей

В качестве объекта переноса знаний была выбрана нейронная сеть AlexNet, решающая эту задачу. Для конфигурации сетей в качестве исходной задачи была выбрана задача классификации изображений, которая содержит 1000 классов изображений.

Натренированная модель содержит 2 блока, которые состоят из свертки, нормализации и max pooling, за ними следуют две свертки, max pooling, flatten и два полносвязных слоя размерности 4096 с функцией активации ReLU, завершается полносвязным слоем с функцией активации softmax, размерности 1000.

## Конфигурация №1 Прямое использование обученной модели

В данном эксперименте использовали натренированную сеть AlexNet, описанную выше.

По результатам использования можно сделать вывод, что данные задачи классификации изображений Quick, Draw! на 5 классов не подходят для предобученной на наборе данных ImageNet модели нейронной сети AlexNet.

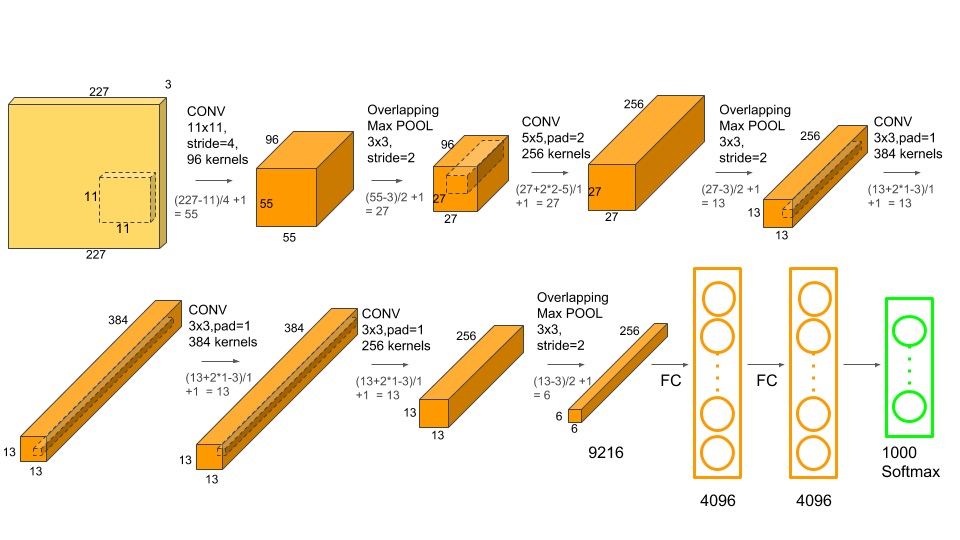
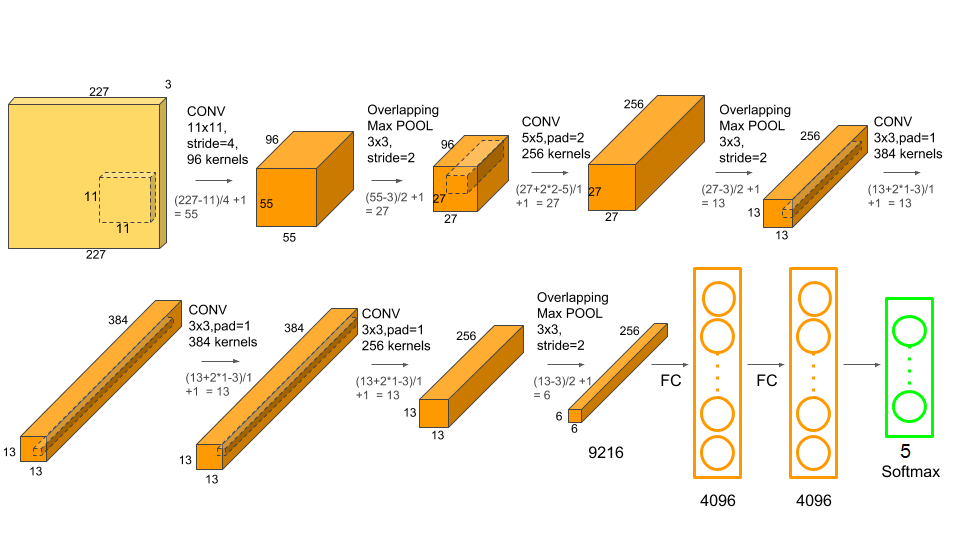


Рис.2 Схема тестовых архитектур сетей

## Конфигурация №2 Использование структуры обученной модели

В данном эксперименте перенесена архитектура натренированной сети AlexNet, где на последнем слое размерность равна 5 – число классов в нашей задаче.

Рис.4 Конфигурация №2

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

**Таблица 2**

**Значения метрик в эксперименте №2**

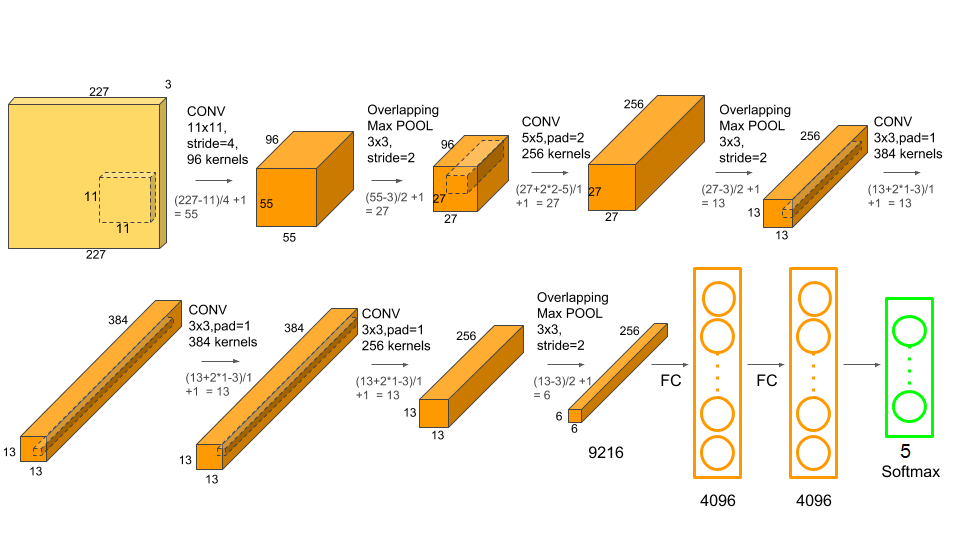
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **precision** | **recall** | **f1-score** | **support** |
| **spoon** | 0.95 | 0.93 | 0.94 | 1015 |
| **paintbrush** | 0.93 | 0.95 | 0.94 | 992 |
| **smiley face** | 0.96 | 0.93 | 0.94 | 1024 |
| **wheel** | 0.93 | 0.94 | 0.93 | 997 |
| **bush** | 0.91 | 0.94 | 0.92 | 972 |
| **micro avg** | 0.92 | 0.92 | 0.91 | 5000 |
| **macro avg** | 0.92 | 0.92 | 0.91 | 5000 |
| **weighted avg** | 0.92 | 0.92 | 0.91 | 5000 |

Из таблицы 2 видно, что показатели достаточно высокие, что говорит об успешном применении метода переноса архитектуры.

## Конфигурация №3 Фиксированный метод извлечения признаков модели

В данном эксперименте перенесены архитектура натренированной сети AlexNet, где на последнем слое размерность равна 5 – число классов в нашей задаче.

Замораживаем обучение всех слоев, кроме двух последних полносвязных слоев.

Рис.5 Конфигурация №3

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

**Таблица 3**

**Значения метрик в эксперименте №3**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **precision** | **recall** | **f1-score** | **support** |
| **spoon** | 0.94 | 0.93 | 0.93 | 1009 |
| **paintbrush** | 0.93 | 0.98 | 0.95 | 998 |
| **smiley face** | 0.94 | 0.93 | 0.93 | 1005 |
| **wheel** | 0.85 | 0.9 | 0.87 | 893 |
| **bush** | 0.95 | 0.77 | 0.85 | 1095 |
| **micro avg** | 0.92 | 0.92 | 0.91 | 5000 |
| **macro avg** | 0.92 | 0.92 | 0.35 | 5000 |
| **weighted avg** | 0.92 | 0.92 | 0.91 | 5000 |

Из таблицы 3 видно, что показатели достаточно высокие, что говорит об успешном применении метода извлечения признаков.

# 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.

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**

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Результаты обучения** | | |
| Точность | Ошибка | Время |
| **1** | 0.01 | 1.58 | 0 мин |
| **2** | 0.99 | 0.10 | 2д 2ч 05 мин |
| **3** | 0.97 | 0.15 | 2ч 45 мин |

Таблица 5 позывает, что перенос архитектуры дает высокую точность, но занимает много времени обучения. Метод извлечения признаков обучается быстрее, но дает меньшую точность, относительно переноса архитектуры.

# Выводы

В ходе лабораторной работы исследовали возможности переноса обучения для решения целевой задачи, выбранной изначально для выполнения практических работ. Искали исходную задачу, близкую по смыслу к целевой задаче, и натренированную модель для решения исходной задачи.

Выполнили три типа экспериментов, описанных выше в отчете, для каждого из которых была сконструирована определенная конфигурация, было произведено обучение модели на выбранном наборе данных задачи [Quick, Draw!](https://quickdraw.withgoogle.com/), провели тестирование программы, в ходе которого собирали метрики качества решения задачи.

Собранные метрики, показатели и результаты были визуализированы с помощью модуля TensorBoard библиотеки TensorFlow.

В ходе экспериментов было установлено, что метод извлечения признаков обучается быстрее, но дает меньшую точность, относительно переноса архитектуры.

Таким образом, цель работы - исследовать возможности переноса обучения для решения целевой задачи, выбранной изначально для выполнения работ - была достигнута.