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 №4

**Начальная настройка весов 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: использовать методы обучения без учителя для настройки начальных значений весов сетей, построенных при выполнении предшествующих практических работ.

# Задачи

Выполнение практической работы предполагает решение следующих задач:

1. Выбор архитектур нейронных сетей, построенных при выполнении предшествующих практических работ.
2. Выбор методов обучения без учителя для выполнения настройки начальных значений весов сетей.
3. Применение методов обучения без учителя к выбранному набору сетей.
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.

## 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\_ae.py - непосредственно обучение сети, использует класс NetworkBase;
* autoencoder.py – описание архитектуры автокодировщика;
* run\_statisctic\_conv.py - получение статистики по итогам обучения.

# Тестовые конфигурации сетей

В ходе лабораторной работы рассмотрели 3 архитектуры сверточных нейронных сетей, реализованных в лабораторной работе №2, для которых реализовали начальную настройку весов с помощью автокодировщиков.

Разработанная сеть представляет из себя систему нескольких блоков, каждый из которых обучается в отдельности. Каждый из блоков состоит из кодирующего и декодирующего части:

* кодировщик – часть (изначальных) слоев нейронной сети,
* декодировщик – набор слоев нейронной сети, реализующих обратные операции кодирующей части.

На вход сеть принимала данные аналогичные, входным данным исходной сети и обучалась тождественному преобразованию полученного входа (ожидаемый выход равен входу). После обучения очередного блока, веса с кодировщика переносятся на слой исходной сети, а выход с кодирующего слоя передается на вход следующему блоку. Процесс повторяется на последующих блоках. После этого процесса запускается обучение последнего полносвязного слоя нейронной сети.

Автокодировщики обучались с шумом, который представляется случайными пикселями на подаваемом изображении, согласно нормальному распределению в некотором соотношении к общему числу пикселей изображения. В экспериментах рассматривались варианты шума в 20%, 40%.

Их структуру можно представить в следующем виде, пример для двух блоков:

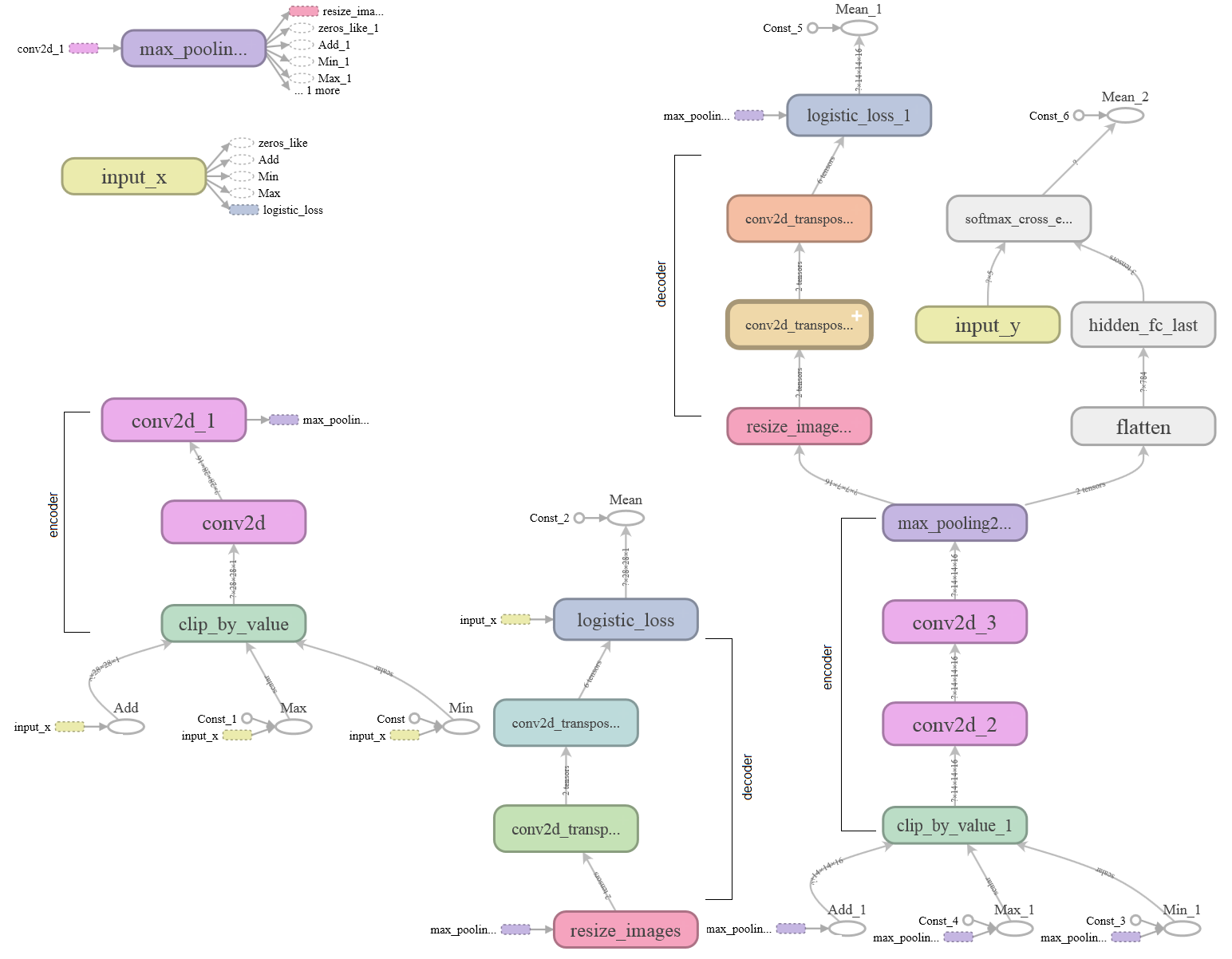


Рис.2 Стек автокодировщиков из двух блоков

В каждом блоке использовались следующие функции:

* функция активации sigmoid:
* функция потерь - cross entropy
* функция активации на последнем слое – softmax.

## Конфигурация №1

Рассматривалась сеть из 2-х блоков:

* каждый блок состоит из
  + 2 свёрточных слоёв,
  + завершается max pooling’ом
  + добавленный шум 20%
* image\_resize по алгоритму ближайших k-соседей;
* Transposed Convolution х2 c ядром 3х3, шаг =1, дополнение границ
* операция пространственного объединения pooling - 3х3, шаг =1
* последний - полносвязный слой состоит из 5 нейронов

Использованы в ходе эксперимента:

* функция активации - ReLU6
* функция потерь - cross entropy
* скорость обучения - 0.001
* оптимизатор – GradientDescentOptimizer
* функция активации на полносвязном слое – softmax

функция активации на полносвязном слое – softmax

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

## Конфигурация №2

Рассматривалась сеть из 2-х блоков:

* каждый блок состоит из
  + 2 свёрточных слоёв,
  + завершается max pooling’ом
  + добавленный шум 40%
* image\_resize по алгоритму ближайших k-соседей;
* Transposed Convolution х2 c ядром 3х3, шаг =1, дополнение границ
* операция пространственного объединения pooling - 3х3, шаг =1
* последний - полносвязный слой состоит из 5 нейронов

Использованы в ходе эксперимента:

* функция активации - ReLU6
* функция потерь - cross entropy
* скорость обучения - 0.001
* оптимизатор – GradientDescentOptimizer
* функция активации на полносвязном слое – softmax

функция активации на полносвязном слое – softmax.

## Конфигурация №3

Рассматривалась сеть из 3-х блоков:

* каждый блок состоит из
  + 2 свёрточных слоёв,
  + завершается max pooling’ом
  + добавленный шум 20%
* image\_resize по алгоритму ближайших k-соседей;
* Transposed Convolution х2 c ядром 3х3, шаг =1, дополнение границ
* операция пространственного объединения pooling - 3х3, шаг =1
* последний - полносвязный слой состоит из 5 нейронов

Использованы в ходе эксперимента:

* функция активации - ReLU6
* функция потерь - cross entropy
* скорость обучения - 0.001
* оптимизатор – GradientDescentOptimizer
* функция активации на полносвязном слое – softmax

## Конфигурация №4

Рассматривалась сеть из 3-х блоков:

* каждый блок состоит из
  + 2 свёрточных слоёв,
  + завершается max pooling’ом
  + добавленный шум 40%
* image\_resize по алгоритму ближайших k-соседей;
* Transposed Convolution х2 c ядром 3х3, шаг =1, дополнение границ
* операция пространственного объединения pooling - 3х3, шаг =1
* последний - полносвязный слой состоит из 5 нейронов

Использованы в ходе эксперимента:

* функция активации - ReLU6
* функция потерь - cross entropy
* скорость обучения - 0.001
* оптимизатор – GradientDescentOptimizer
* функция активации на полносвязном слое – softmax

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;

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** | **Шум** | **Результаты обучения** | | |
| Точность | Значение функции ошибки (cross entropy) | Время |
| **1** | 2 | 20% | 0.76 | 0.41 | 1ч 17 мин |
| **2** | 2 | 40% | 0.84 | 0.24 | 1ч 21 мин |
| **3** | 3 | 20% | 0.85 | 0.17 | 1ч 38 мин |
| **4** | 3 | 40% | 0.97 | 0.18 | 1ч 47 мин |

Таблица 5 позывает, что наилучший результат достигается при конфигурации №4, при этом точность достигает отметки в 0.97

# Выводы

В ходе лабораторной работы выбирали архитектуры нейронных сетей, построенных при выполнении предшествующих практических работ, затем выбирали различные методы обучения без учителя для выполнения настройки начальных значений весов нейронных сетей.

На следующем этапе работы применяли выбранные методы обучения без учителя к выбранному набору конфигураций сетей, разработали программу, применяющую выбранные конфигурации и методы к набору 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, полученные графики представлены выше в данном отчёте.

В ходе экспериментов было установлено, что наилучший результат дости достигается при конфигурации №4, при этом точность достигает отметки в 0.97.

Таким образом, цель работы - использовать методы обучения без учителя для настройки начальных значений весов сетей, построенных при выполнении предшествующих практических работ - была достигнута.