

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
Национальный исследовательский Нижегородский государственный университет
им. Н.И. Лобачевского

Институт информационных технологий, математики и механики

Кафедра математического обеспечения и суперкомпьютерных технологий

Направление подготовки
02.04.02. Фундаментальная информатика и информационные технологии

Направленность образовательной программы
магистерская программа «Компьютерная графика и моделирование живых и
технических систем»

Отчёт
по методам глубокого обучения для решения задач компьютерного зрения

на тему:
«Разработка сверточных нейронных сетей»

Квалификация (степень)
магистр

Форма обучения
очная

Выполнили: студенты группы 381706-3М
Храмов Илья Валерьевич

Подпись

Реунова Ольга Алексеевна

Подпись

Воеводин Андрей Михайлович

Подпись

Н. Новгород
2018 г.

Содержание

Постановка задачи.....	3
Тренировочные и тестовые наборы данных.....	4
Метрика качества решения.....	6
Разработанные программы/скрипты.....	7
Тестовые конфигурации сетей.....	8
Результаты.....	9
Литература.....	12

Постановка задачи

Цель

Цель настоящей работы состоит в том, чтобы 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 выбранной библиотекой глубокого обучения.
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.

Тренировочные и тестовые наборы данных

Задача — классификация комиксов. Данные получены из [5]. 86 классов сокращены до 14 классов с целью убрать классы, в которых выборка не репрезентативна. Размер изображений в каждом классе 288*432. Изображения 3 каналные.

№	Категории	Размер тренировочной выборки	Размер тестовой выборки
1	Aquaman v7	1088	282
2	Batgirl v4	1088	293
3	Batman v2	1509	384
4	Batwing	616	167
5	Batwoman	739	187
6	Catwoman v4	1047	256
7	Green Arrow	1040	256
8	Green Lantern	1301	345
9	Harley Quinn	671	165
10	Nightwing v3	558	147
11	Red Lanterns	767	197
12	Sinestro	619	143
13	Supergirl v6	701	200
14	Wonder Woman	1233	292
		12977	3314

Таблица 1. Размер выборки в каждом классе тренировочного и тестового множеств.

Изображения хранятся в формате JPEG.

На вход сети подаются бинарные файлы с расширениями .rec (изображения), .idx (индексы изображений) (Рис. 1).

Binary Record

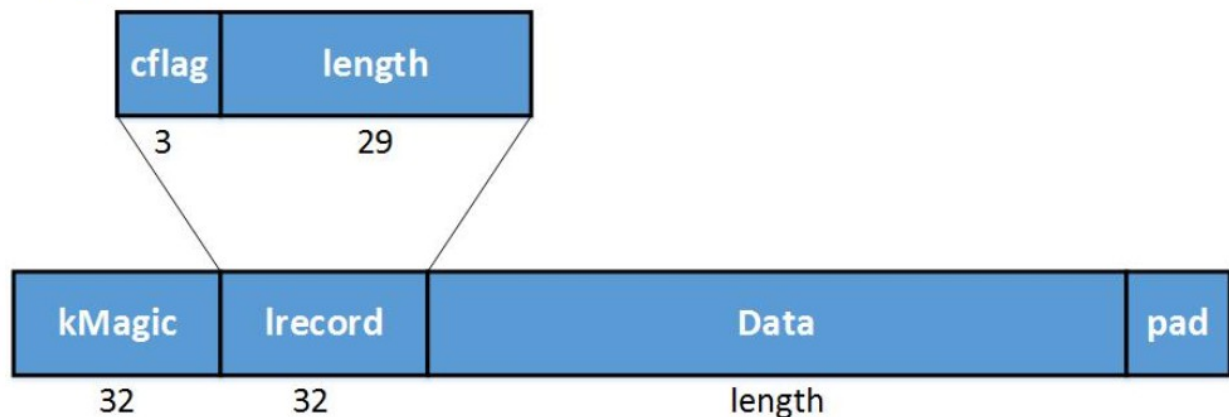


Рисунок 1. Формат хранения в MXNet [8].

- kMagic – начало записи;
- lrecord – длина (length) и продолжительность записи (cflag);
- Data – данные;
- pad – пространство для выравнивания до 4 байт.

Метрика качества решения

В качестве метрики для оценки качества решения задачи [5] выбрана “Точность” (“Accuracy”). В терминологии MXNet — это отношение количества правильно предсказанных сэмплов к общему количеству сэмплов (Рис. 2).

$$\text{accuracy}(y, \hat{y}) = \frac{1}{n} \sum_{i=0}^{n-1} 1(\hat{y}_i == y_i)$$

Рисунок 2. Определение “Accuracy” в MXNet [7].

Более подробную информацию о метриках можно найти в [7].

Разработанные программы/скрипты

- `reduce_dataset.py` - скрипт для вычленения из оригинального набора данных наиболее репрезентативных категорий.
- `prepare_dataset.py` - скрипт для подготовки данных под `mxnet`.
- `lab3.py` - скрипт для обучения сверточных нейронных сетей.
- `resize_dataset.py` - скрипт для изменения размерности данных.
- `load_dataset.py` - функции для загрузки данных в скрипт обучения.
- `blocks.py` - основные конструкционные блоки сетей.
- `fit.py` - универсальный скрипт обучения, формирующий все необходимые отчёты.
- `parse_log.py` - скрипт, формирующий сводную таблицу из журнала обучения.

Тестовые конфигурации сетей

1.

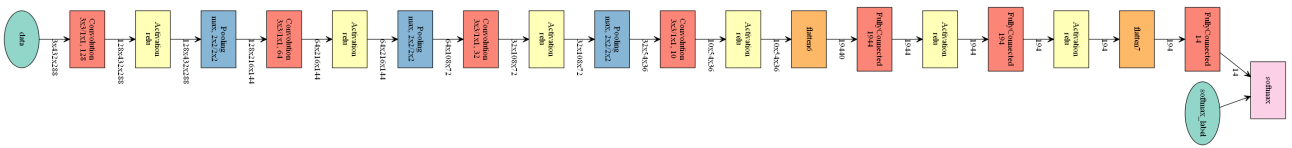


Рисунок 3. Сеть1.

2.

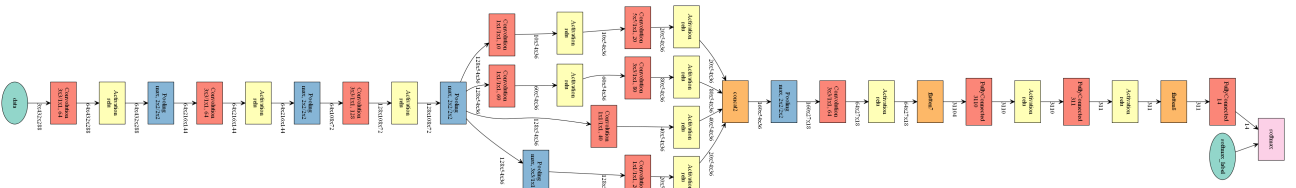


Рисунок 4. Сеть2.

3.

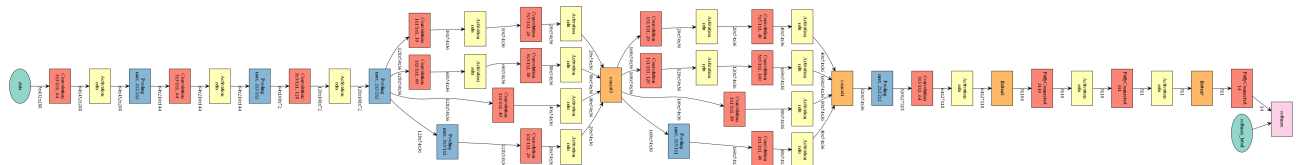


Рисунок 5. Сеть3.

4.

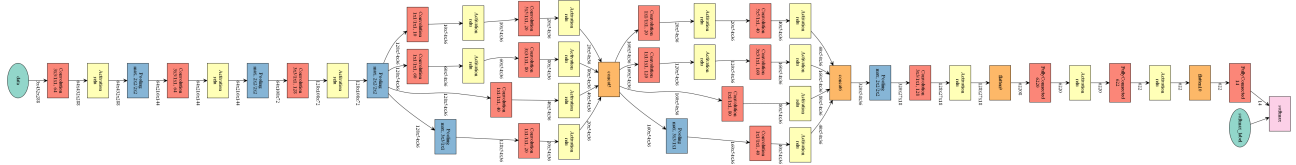


Рисунок 6. Сеть4.

5.

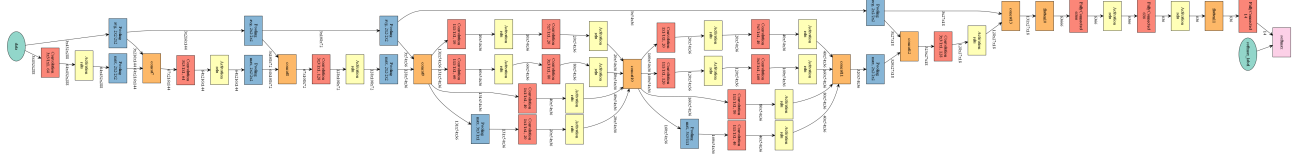


Рисунок 7. Сеть5.

Результаты

В таблице 2 приведены конфигурация системы и программное обеспечение, с помощью которых проводилось обучение и тестирование построенных моделей.

Параметры	Версия
Операционная система	Windows10
GPU	NVIDIA GeForce GTX 1080 (частота процессора — OC Mode – GPU Boost Clock : 1835 MHz , GPU Base Clock : 1695 MHz Gaming Mode (Default) - GPU Boost Clock : 1809 MHz , GPU Base Clock : 1670 MHz; шина передачи данных – PCI Express 3.0; видеопамять - GDDR5X 8GB; количество ядер — 2560; частота памяти - 10010 MHz; интерфейс памяти — 256-bit;)
CUDA	9.2
Python	3.7.1
MXNet	1.3.0

Таблица 2. Конфигурация системы.

В таблице 3 приведены параметры обучения.

Наименование сети	Оптимизатор	Скорость обучения	Количество эпох	Размер batch
Сеть1	SGD (стохастический градиентный спуск)	0.0001	60	10
Сеть2	SGD	0.0001	60	10
Сеть3	SGD	0.0001	60	10
Сеть4	SGD	0.0001	60	10
Сеть5	SGD	0.0001	60	10

Таблица 3. Параметры обучения.

	Наименование сети				
	Сеть1	Сеть2	Сеть3	Сеть4	Сеть5
Среднее время обучения за одну эпоху, с	114,72	94,02	108,98	156,26	171
Качество решения на тренировочном наборе (Ассигасу), %	93,23	93,96	93,65	93,85	88,85
Качество решения на тестовом наборе (Ассигасу), %	18,55	28,76	30	31,33	23,14
Номер эпохи с достигнутым максимальным качеством решения на тренировочном наборе	48	48	48	46	58
Максимальное качество решения на тренировочном наборе (Ассигасу), %	94,95	96,39	95,88	96,08	91,55
Номер эпохи с достигнутым максимальным качеством решения на тестовом наборе	57	43	56	48	58
Максимальное качество решения на тестовом наборе (Ассигасу), %	19,91	31,36	31,72	34,13	24,46

Таблица 4. Результаты экспериментов. Конфигурация сетей приведена в “Тестовые конфигурации сетей”.

Анализ результатов

Из результатов, представленных в таблице 4, можно сделать следующие выводы:

1. Выборка — нерепрезентативная. Малое количество изображений на один класс, как тренировочной, так и тестовой выборок. Некоторые изображения в тестовой и тренировочной выборках слишком отличаются (например, тренировочная выборка — нарисован Бэтман на черном фоне, тестовая выборка — нарисован Бэтман на светлом фоне). Герои одних комиксов могут встречаться в других комиксах (например, Harley Quinn в Batman и наоборот).

2. Увеличение “глубины” сети не даёт сильный прирост точности на тестовой выборке.
3. Наблюдается переобучение сети (например, сеть 2, 3).
4. Для разрешения низкой точности на тестовой выборке, рекомендуется использовать архитектуру Inception.

Литература

1. MNIST dataset [<http://yann.lecun.com/exdb/mnist>].
2. OpenCV [<http://opencv.org>].
3. Материалы Летней межвузовской школы 2016 [<https://github.com/itseez-academy/itseez-ss-2016-theory>], [<https://github.com/itseez-academy/itseez-ss-2016-practice>].
4. Лекции по глубокому обучению: <https://sites.google.com/site/kustikovavalentina/studentam/kurs-glubokoe-obucenie>, 2018.
5. Исходные данные — <https://www.kaggle.com/cenkbircanoglu/comic-books-classification>: kaggle datasets download -d cenkbircanoglu/comic-books-classification.
6. Документация MXNet — <http://mxnet.incubator.apache.org/test/tutorials/>.
7. Метрики в MXNet – <https://mxnet.incubator.apache.org/api/python/metric/metric.html>.
8. Формат хранения данных в MXNet: https://mxnet.incubator.apache.org/architecture/note_data_loading.html.
9. Репозиторий исходных кодов: <https://github.com/okondratieva/DeepLearning>.