Волгоградский государственный университет
Институт математики и информационных технологий
Кафедра информационных систем и компьютерного моделирования

Работа допущена к защите
Заведующий кафедрой

_____ А. В. Хоперсков

«____» _____ 2021 г.

Борисовский Егор Иванович

Нейросетевая регрессия экспериментальных данных

Отчет по учебной практике, научно-исследовательской работе

Направление: 09.04.01 – Информатика и вычислительная техника

Студент группы ИВТм-201:
Е. И. Борисовский _____ подпись

Руководитель практики:
С. С. Храпов, к.ф-м.н., доцент каф. ИСКМ _____ подпись

Ответственный за организацию практики:
А. В. Хоперсков, д.ф.-м.н., профессор каф. ИСКМ _____ подпись

Волгоград — 2021

Содержание

Введение

В последние несколько лет в различных сферах жизни человека очень часто используются алгоритмы машинного обучения и нейросети. Они помогают решать совершенно разные задачи - от подбора персональных рекомендаций для просмотра фильмов, до помощи в диагностировании различных заболеваний на ранней стадии. В процессе разработки программного обеспечения с подобным функционалом происходит очень много итераций по настройке модели и подбору параметров для нее. Эти рутинные операции в основном производятся с помощью замены обучающей выборки, изменения значений параметров в коде и перезапуска программы. На некоторых этапах разработки может потребоваться показать свои результаты другому человеку, возможно далекому от деталей реализации получившейся нейросетевой модели и появляется необходимость в простом универсальном пользовательском интерфейсе, который можно было бы к ней подключить и использовать.

В данной работе рассматриваются области применения нейросетей, популярные библиотеки с их реализацией и существующие в данный момент решения на рынке для подключения к ним графического интерфейса.

Главной целью работы является проектирование и разработка библиотеки с реализацией пользовательского графического интерфейса для задач машинного обучения и нейросетей на языке Python. Так же необходимо оформить получившуюся программу в pip-пакет и опубликовать его в сервисе PyPI.

Глава 1

Методы и области применения машинного обучения и нейросетей

1.1 Области применения

Использование алгоритмов машинного обучения и нейросетей позволяет решать задачи в различных сферах деятельности человека, таких как недвижимость, сельское хозяйство, экономика, а так же медицина. По данным агенства Frost & Sullivan спрос на разработки, в которых используется машинное обучение в медицине, увеличивается с каждым годом примерно на 40% [**habrbigdatamedicine**]. Такие разработки могут использоваться как для диагностики заболеваний, так и для биохимических исследований.

Методы машинного обучения активно применяются при медицинском сканировании различных типов, таких как УЗИ или компьютерная томография. Благодаря алгоритмам распознавания образов на изображениях есть возможность анализировать результаты таких исследований и указывать на проблемные участки. Также возможно определение диагноза пациента по различным его параметрам и результатам исследования. Но программное обеспечение, использующее данные алгоритмы пока не может заменить полностью работу медиков и используется в основном при первичных исследованиях в качестве экспертных систем.

При компьютерном моделировании алгоритмы машинного обучения могут использоваться для валидации получившихся данных, или прогнозирования течения каких-либо физических процессов.

1.2 Популярные библиотеки с реализацией нейронных сетей

На текущий момент существует множество готовых реализаций нейросетей и алгоритмов машинного обучения, что не имеет смысла делать то же самое с нуля, если задача не имеет каких-то особенностей, делающих невозможным использование готовых библиотек. Каждая из библиотек, рассматриваемых в работе, хороша в своей области, успешно используется в решении задач и проверена временем. Рассмотрим некоторые из популярных библиотек для языка программирования Python по данным рейтинга на GitHub (рисунок 1.1)

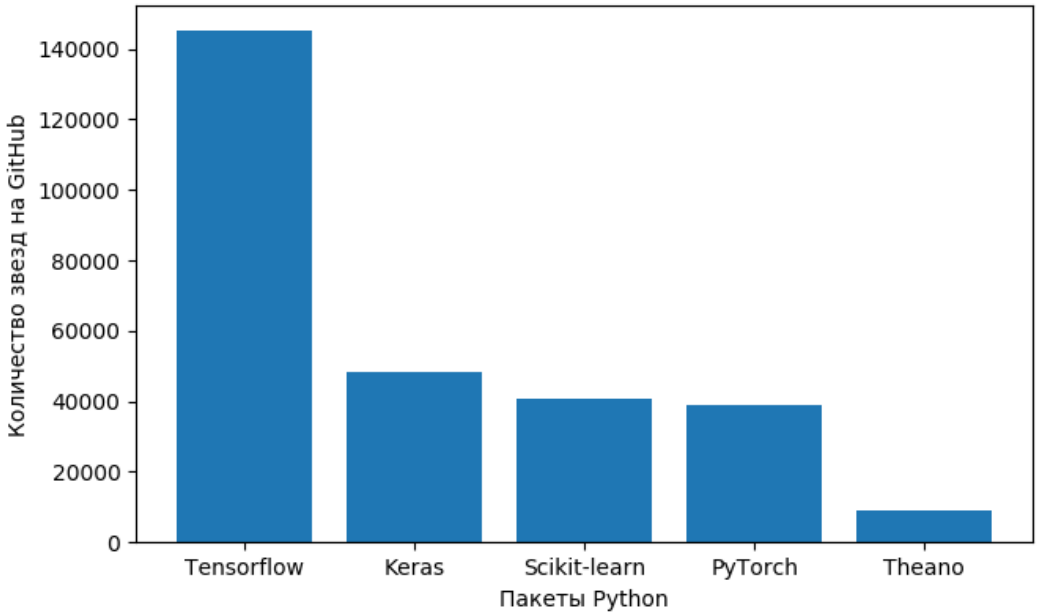

Рисунок 1.1. Популярные пакеты Python для машинного обучения по данным рейтинга на GitHub

1.2.1 TensorFlow

Самой популярной и масштабной по применению является библиотека TensorFlow, используемая для глубокого машинного обучения [**gudfellow**].

Библиотека разрабатывается в тесном сотрудничестве с компанией Google и применяется в большинстве их проектов где используется машинное обучение. Библиотека использует систему многоуровневых узлов, которая позволяет вам быстро настраивать, обучать и развертывать искусственные нейронные сети с большими наборами данных.

Библиотека хорошо подходит для широкого семейства техник машинного обучения, а не только для глубокого машинного обучения. Программы с использованием TensorFlow можно компилировать и запускать как на CPU, так и на GPU. Также данная библиотека имеет обширный встроенный функционал логирования, собственный интерактивный визуализатор данных и логов [**muller**].

1.2.2 Keras

Keras используется для быстрого прототипирования систем с использованием нейронных сетей и машинного обучения. Пакет представляет из себя высокоуровневый API, который работает поверх TensorFlow или Theano. Поддерживает как вычисления на CPU, так и на GPU

1.2.3 Scikit-learn

Scikit-learn – это одна из самых популярных библиотек для языка Python, в которой реализованы основные алгоритмы машинного обучения, такие как классификация различных типов, регрессия и кластеризация данных. Библиотека распространяется свободно и является бесплатной для использования в своих проектах [**rashka**].

Данная библиотека создана на основе двух других – NumPy и SciPy, имеющих большое количество готовых реализаций часто используемых математических и статистических функций. Библиотека хорошо подходит для простых и средней сложности задач, а также для людей, которые только

начинают свой путь в изучении машинного обучения.

1.2.4 PyTorch

PyTorch – это популяный пакет Python для глубокого машинного обучения, который можно использовать для расширения функционала совместно с такими пакетами как NumPy, SciPy и Cython. Главной функцией PyTorch является возможность вычислений с использованием GPU. Отличается высокой скоростью работы и удобным API-интерфейсом расширения с помощью своей логики, написанной на C или C++.

1.2.5 Theano

Theano – это библиотека, в которой содержится базовый набор инструментов для машинного обучения и конфигурирования нейросетей. Так же у данной библиотеки есть встроенные методы для эффективного вычисления математических выражений, содержащих многомерные массивы [**rashka**].

Theano тесно интегрирована с библиотекой NumPy, что дает возможность просто и быстро производить вычисления. Главным преимуществом библиотеки является возможность использования GPU без изменения кода программы, что дает преимущество при выполнении ресуркоемких задач. Также возможно использование динамической генерации кода на языке программирования C [**douson**].

1.2.6 Обзор существующих решений реализации графического интерфейса для задач машинного обучения

Прежде чем разрабатывать приложение из данной работы была произведена попытка найти существующие готовые решения для текущей задачи. Были найдены всего лишь два решения: проект на GitHub MachineLearningGUI и программный комплекс Weka. Рассмотрим каждое из

них отдельно.

MachineLearningGUI – десктопное приложение, написанное на языке программирования Python с помощью библиотеки PyQt. Работает только с библиотекой Scikit-learn и алгоритмом классификации с деревом принятия решений. Так же автор проекта указал в описании, что приложение работает только с одним набором данных, который идет вместе с проектом. Интерфейс программы состоит из четырех вкладок, каждая из которых отвечает за конкретный шаг: загрузка обучающей выборки, препроцессинг данных, запуск алгоритма и просмотр результатов. На рисунках ?? и ?? представлены первый и третий шаги.

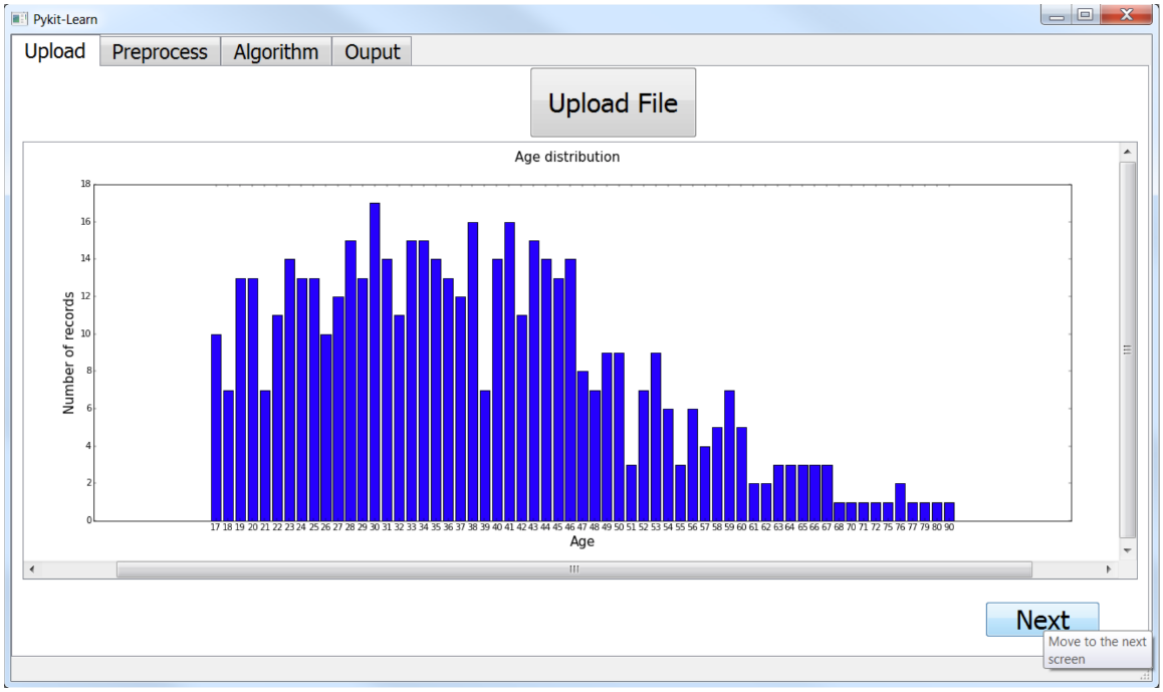

Рисунок 1.2. Интерфейс первого шага с загрузкой файла обучающей выборки

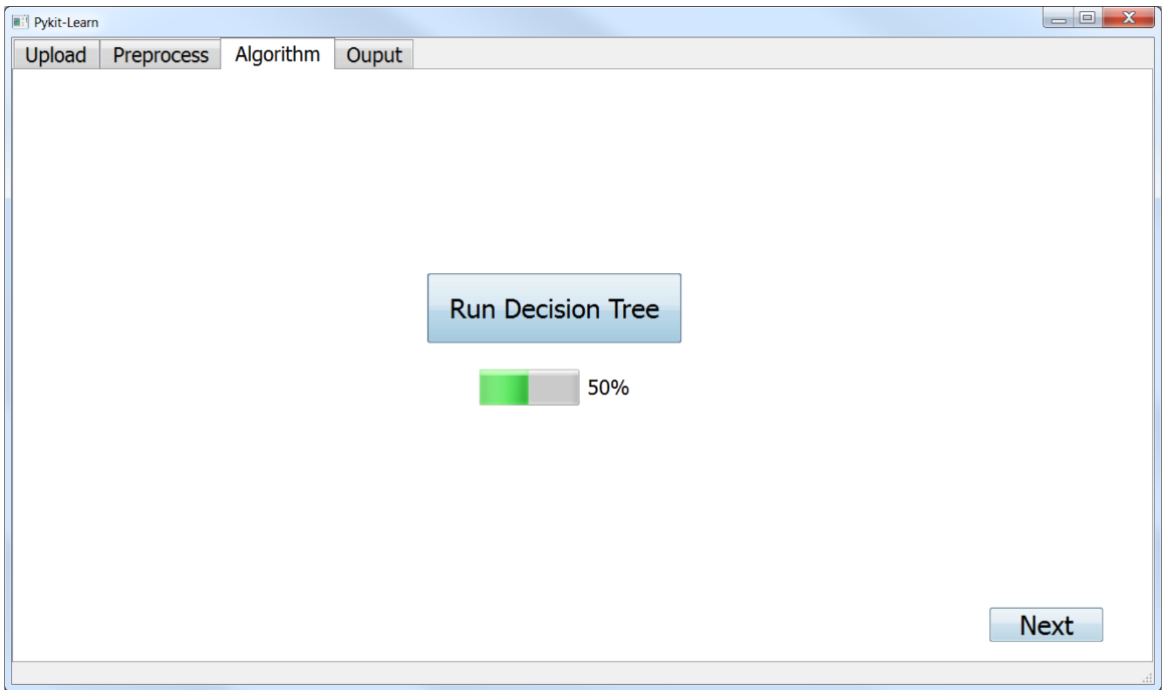

Рисунок 1.3. Интерфейс третьего шага с запуском алгоритма

В целом проект выглядит сыро и не кажется пригодным для использования, по крайней мере для задачи из данной работы.

Weka – открытый программный комплекс, содержащий в себе реализации алгоритмов машинного обучения для решения задач интеллектуального анализа. Проект разработан на языке программирования Java на базе университета Вайкато в Новой Зеландии. Целью проекта является создание современной среды для разработки и применения методов машинного обучения к реальным данным и упрощения этого процесса. Weka широко используется в учебных целях и исследователями в области машинного обучения. В состав комплекса входят средства для препроцессинга данных, классификации, регрессии, кластеризации и визуализации результатов[**weka1**].

Для работы в Weka необходимо загрузить файл с обучающей выборкой. На вкладке Preprocess (рисунок **??**) можно увидеть статистические метрики, расчитанные по выборки и применить один или несколько фильтров к набору данных.

Глава 2

Элементы, которые могут присутствовать в тексте

2.1 Формулы

Для оформления формул и уравнений используются стандартные возможности LaTeX. Выключные формулы должна быть оформлены как окружение `equation`. Пояснения символов и числовых коэффициентов, входящих в формулу, если они не пояснены ранее в тексте, должны быть приведены непосредственно под формулой. Пояснение каждого символа следует давать в той последовательности, в которой символы приведены в формуле. Первая строка пояснения должна начинаться со слова «где» без двоеточия после него. Формулы, следующие одна за другой и не разделенные текстом, разделяют запятой

Ниже приведен простейший пример

$$H_0 = \frac{p^2}{2m} + a_0\,x^4 - a_2\,x^2, \tag{2.1}$$

где p – импульс частицы, m – масса частицы, x – координата частицы, a_0, a_2 – параметры системы.

Нумерация формул осуществляется автоматически в пределах главы, как это продемонстрировано для формулы (2.1). Переносить формулы на следующую строку допускается только на знаках выполняемых операций, причем знак в начале следующей строки повторяют.

Ссылки в тексте на порядковые номера формул дают в скобках, например, . . . в формуле (2.1).

А это уже более сложный пример

$$H^0_{m\,n} = \delta_{m+4\,n}\,\frac{a_0 g^2}{4}\sqrt{(m+1)(m+2)(m+3)(m+4)} + \\ + \delta_{m+2\,n}\,\frac{g}{2}a_0(2m+3)\sqrt{(m+1)(m+2)}. \qquad (2.2)$$

где $g = \frac{m}{a_0}$ – числовой коэффициент, или так $g = \dfrac{m}{a_0}$, если поместить внутристрочную формулу в аргумент команды `\displaystyle{}`.

2.2 Рисунки

Для вставки рисунков рекомендуется использовать пакеты graphics или graphicx и команду `\includegraphics`. Рисунок оформляется с использованием стандартного окружения `figure`, которое обеспечивает его автоматическую нумерацию. Рисунок и подпись вставляются при помощи стандартных команд `\includegraphics` и `\caption{Подпись}` соответственно. Подрисуночная подпись должна пояснять что изображено на рисунке, также может содержать уточняющую информацию.

Рисунки должны быть пронумерованы и иметь подпись, которая всегда должна располагаться под рисунками. Рисунки могут быть представлены в формате * .pdf, * .ps, * .eps, * .jpg, * .png или * .tif и должны иметь разрешение не менее 300 dpi.

Убедитесь, что линии на рисунках не прерываются и имеют постоянную ширину. Сетки и детали на рисунках должны быть четкими и не должны быть начертаны друг на друге. Аббревиатуры, используемые на рисунке, должны быть определены в тексте отчета, если они не являются общими сокращениями или уже были определены в тексте. Легенда должна пояснять все используемые символы и должна входить с состав рисунке, а не содержаться в словесных пояснениях в подписях (например, «пунктирная линия» или «открытые зеленые кружки»).

На все рисунки в тексте должна быть ссылка до появления самого рисунка в тексте отчета, например: рисунки **??**, 2.1.

Рисунок 2.1. Пример использования *.jpg

Если возникает необходимость привести в тексте рисунок, который не помещается на одной странице, например, это может быть большая блок-схема, то номер рисунка не должен изменяться, а в конце подписи к нему должно быть указано: Лист 1. На следующей странице должен быть приведен только номер рисунка, а подпись должна содержать только Лист 2. Например: Рисунок 2.1 – Лист 2.

2.3 Таблицы

Для создания таблиц используется окружение `table`, которое обеспечивает нумерацию и создание заголовка. Для помещения заголовка над таблицей в начале указанного окружения необходимо задать команду `\capition{Подпись к таблице}`. Сама таблица может оформляться с помощью стандартного окружения `tabular`. Пример таблицы приведен ниже (таблица 2.1).

На все таблицы документа должны быть приведены ссылки в тексте документа до появления самой таблицы. При ссылке следует писать слово

Таблица 2.1. Название таблицы. Таблицы следует размещать в основном тексте рядом с первым цитированием.

Заголовок 1	Заголовок 2	Заголовок 3
Запись 1	данные	данные
Запись 2	данные	данные
Запись 3	данные	данные

«таблица» с указанием её номера.

В случае наличия в тексте длинных таблиц, которые не помещаются на одной странице, необходимо использовать окружение `longtable`.

2.4 Листинги программ

Листинги программ могут быть расположены как в тексте отчета (возможно ближе к соответствующим частям текста, содержащим ссылку на листинг), так и в конце его текста отчета (в приложениях). Листинг, за исключением листингов приложений, следует нумеровать арабскими цифрами с нумерацией в пределах главы (листинг 2.1). Листинг каждого приложения обозначают отдельной нумерацией арабскими цифрами с добавлением перед цифрой обозначения приложения. Например: Листинг А.1. На все листинги должны быть ссылки в тексте отчета до появления самого листинга, примеры таких ссылок приведены выше.

Листинг 2.1. Пример листинга в тексте

```
1 // a simple kernel that simply increments each array element by b
2 __global__ void kernelAddConstant(int *g_a, const int b) {
3   int idx = blockIdx.x * blockDim.x + threadIdx.x;
4   g_a[idx] += b;
5 }
```

Глава 3

Оформление библиографии

Для оформления списка литературы в шабоне используется система BibTEX. Программное обеспечение для создания форматированных списков библиографии BibTEX, позволяет упростить и (полу)автоматизировать процесс оформления списка литературы.

Для использования BibTEX необходимо записать список источников, на которые вы ссылаетесь в своем тексте, в специальный файл с расширением .bib (Bib.bib). BibTEX файл состоит из записей, которые начинаются с символа `@` и указанием типа записи. Что бы ни было записано в вашем bib-файле, BibTEX по умолчанию включает в список литературы только те источники, на которые вы ссылаетесь с помощью команды `\cite`. Использовать команду `\nocite` для включения записей, на которые нет ссылок в тексте, запрещается. Пример библиографической записи [3] в bib-файле приведен в листинге 3.1.

Листинг 3.1. Пример библиографической записи

```
@article{Jackson2021TheOO,
  title={The origin of low-surface-brightness galaxies in the
  dwarf regime},
  Ryan A. Jackson and Garreth Martin and Sugata Kaviraj
  and Marius Ramsoy and Julien Devriendt and Thomas M. Sedgwick
  and C. Laigle and H. Choi and Ricarda S Beckmann and Marta
  Volonteri and Yohan Dubois and Christophe Pichon
  and  Sukyoung K. Yi and Adrianne D. Slyz and Katarina Kraljic
  and Taysun Kimm and S{\'e}bastien Peirani and Ivan K Baldry},
  journal={Monthly Notices of the Royal Astronomical Society},
  year={2021},
  volume={502},
  pages={4262-4276}
```

```
14 }
```

В первой строке, сразу после `@article{`, стоит уникальная библиографическая метка `Jackson2021TheOO`; при ссылках в тексте на указанную работу необходимо написать `\cite{Jackson2021TheOO}`. Далее идут поля записи, также разделенные запятыми Смысл большинства полей ясен из их названия.

Для заполнения `bib`-файла можно воспользоваться библиографическими базами данных, которые позволяют выгрузить готовую запись в `bib`-файл. Пример такой базы и экспорта цитирований приведен на рисунке 3.1

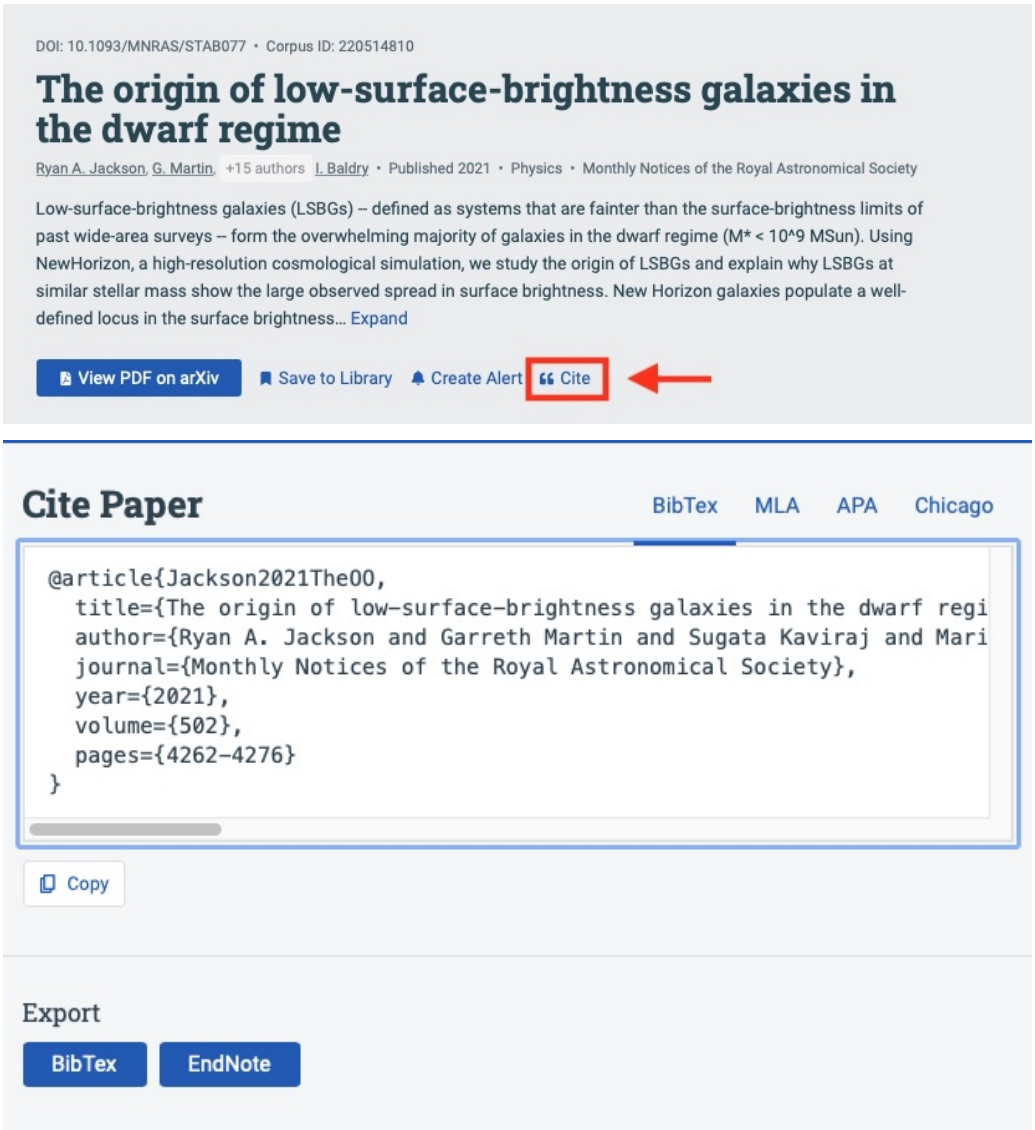

Рисунок 3.1. Пример экспорта BibTEX-файла на примере системы SemanticScholar

ВНИМАНИЕ!

К списку источников предьявляютя особые требования. Это касается как его содержания, так и количества источников, на которые вы ссылаетесь в своем отчете. В первой строке таблицы 3.1 указан номер курса, для которого в строке ниже приведено минимальное общее количество источников, на которые есть ссылки в работе. В последующих строках приведено минимальное количество источников разных типов. Под научно-периодической литературой подразумевается наличие ссылок на статьи в периодических тематических журналах.

Таблица 3.1. Название таблицы. Таблицы следует размещать в основном тексте рядом с первым цитированием.

Номер курса	2	3	4	5	6
Общее количество используемой литературы из них:	> 15	> 20	> 25	> 30	> 30
- на иностранных языках	≥ 4	≥ 5	≥ 6	≥ 7	≥ 7
- текущая научно-периодическая литература (после 2010 г.)	≥ 3	≥ 5	≥ 7	≥ 7	≥ 7
- литература 21 века	≥ 10	≥ 15	≥ 21	≥ 21	≥ 21

Заключение

Заключение должно содержать перечисление результатов, полученных при выполнении работы, а также те выводы, которые вы сделали при ее выполнении. Также заключение может содержать предложения, рекомендации и перспективы дальнейшего развития темы.

Далее в заключении приводится перечень компетенций, освоенных вами за время выполнения практики. Для каждой компетенции приводится её формулировка и описание того, как именно вы ее освоили при выполнении своей работы.Перечень компетенций необходимо взять из Листа задания, который вы получаете от ответственного за организацию практики (вашего научного руководителя).

Список литературы

1. *Cohen P. J.* The independence of the continuum hypothesis // Proceedings of the National Academy of Sciences. — 1963. — Т. 50, № 6. — С. 1143—1148.
2. *Боресков А., Харламов А.* Основы работы с технологией CUDA. — «ДМК Пресс»., 2010 – 232 с.
3. The origin of low-surface-brightness galaxies in the dwarf regime / R. A. Jackson, G. Martin, S. Kaviraj, M. Ramsoy, J. Devriendt, T. M. Sedgwick, C. Laigle, H. Choi, R. S. Beckmann, M. Volonteri, Y. Dubois, C. Pichon, S. K. Yi, A. D. Slyz, K. Kraljic, T. Kimm, S. Peirani, I. K. Baldry // Monthly Notices of the Royal Astronomical Society. — 2021. — Т. 502. — С. 4262—4276.

Приложение А

Листинг разработанной программы

Листинг А.1. Пример листинга в приложении

```
1 /**
2  * Program main
3  */
4 int main(int argc, char **argv) {
5   printf("[Matrix Multiply Using CUDA] - Starting...\n");
6
7   if (checkCmdLineFlag(argc, (const char **)argv, "help") ||
8       checkCmdLineFlag(argc, (const char **)argv, "?")) {
9     printf("Usage -device=n (n >= 0 for deviceID)\n");
10    printf("-wA=WidthA -hA=HeightA (Width x Height of Matrix A)\n
        ");
11    printf("-wB=WidthB -hB=HeightB (Width x Height of Matrix B)\n
        ");
12    printf("Note: Outer matrix dimensions of A & B matrices" \
13           "must be equal.\n");
14
15    exit(EXIT_SUCCESS);
16  }
17
18  // This will pick the best possible CUDA capable device,
       otherwise
19  // override the device ID based on input provided
20  // at the command line
21  int dev = findCudaDevice(argc, (const char **)argv);
22
23  int block_size = 32;
24
25  dim3 dimsA(5 * 2 * block_size, 5 * 2 * block_size, 1);
26  dim3 dimsB(5 * 4 * block_size, 5 * 2 * block_size, 1);
```

```
27
28  // width of Matrix A
29  if (checkCmdLineFlag(argc, (const char **)argv, "wA")) {
30    dimsA.x = getCmdLineArgumentInt(argc, (const char **)argv, "
        wA");
31  }
32
33  // height of Matrix A
34  if (checkCmdLineFlag(argc, (const char **)argv, "hA")) {
35    dimsA.y = getCmdLineArgumentInt(argc, (const char **)argv, "
        hA");
36  }
37
38  // width of Matrix B
39  if (checkCmdLineFlag(argc, (const char **)argv, "wB")) {
40    dimsB.x = getCmdLineArgumentInt(argc, (const char **)argv, "
        wB");
41  }
42
43  // height of Matrix B
44  if (checkCmdLineFlag(argc, (const char **)argv, "hB")) {
45    dimsB.y = getCmdLineArgumentInt(argc, (const char **)argv, "
        hB");
46  }
47
48  if (dimsA.x != dimsB.y) {
49    printf("Error: outer matrix dimensions must be equal.
50             (%d != %d)\n", dimsA.x, dimsB.y);
51    exit(EXIT_FAILURE);
52  }
53
54  printf("MatrixA(%d,%d), MatrixB(%d,%d)\n", dimsA.x, dimsA.y,
55         dimsB.x, dimsB.y);
56
57  int matrix_result = MatrixMultiply(argc,
```

```
58                                          argv,
59                                          block_size,
60                                          dimsA,
61                                          dimsB);
62
63    exit(matrix_result);
64 }
```