Министерство науки и высшего образования РФ
ФГБОУ ВО «Ульяновский государственный университет»
Факультет математики, информационных и авиационных технологий

Кафедра телекоммуникационных технологий и сетей

Липатова Светлана Валерьевна

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ
для семинарских (практических) занятий, лабораторного практикума
и самостоятельной работы
по дисциплине

«Алгоритмы искусственного интеллекта на Python»

для студентов направления
11.04.02 "Инфокоммуникационные технологии и системы связи"

Ульяновск
2022

Методические рекомендации для семинарских (практических) занятий, лабораторного практикума и самостоятельной работы по дисциплине «Алгоритмы искусственного интеллекта на Python» / составитель: С.В. Липатова - Ульяновск: УлГУ, 2022 – 107 с.

Настоящие методические рекомендации предназначены для студентов направления обучения 11.04.02 " Инфокоммуникационные технологии и системы связи". В работе приведены литература по дисциплине, темы дисциплины и вопросы в рамках каждой темы, рекомендации по изучению теоретического материала, контрольные вопросы для самоконтроля, задания для самостоятельной работы, задачи и упражнения для самостоятельной подготовки к семинарам или полностью самостоятельного освоения практических навыков, задания для лабораторного практикума и рекомендации по их выполнению.

Студентам всех форм обучения следует использовать данные методические рекомендации при подготовке к семинарам, самостоятельной подготовке, а также промежуточной аттестации по дисциплине «Алгоритмы искусственного интеллекта на Python».

Рекомендованы к введению в образовательный процесс

Учёным советом факультета математики, информационных и авиационных технологий УлГУ

протокол № 3/19 от «19» апреля 2022 г.

СОДЕРЖАНИЕ

ОБЩИЕ ВОПРОСЫ

В результате изучения дисциплин «Алгоритмы искусственного интеллекта на Python» студенты должны:

– сформировать системное базовое представление, первичные знания, умения и навыки студентов по основам инженерии знаний и нейроинформатики,
– получить общее представление о прикладных системах искусственного интеллекта,
– получить представление о роли искусственного интеллекта и машинного обучения в развитии информатики в целом, а также, в научно-техническом прогрессе,
– подготовиться к применению концепций интеллектуальных систем при дальнейшем обучении,
– подготовиться к научной и практической деятельности в области разработки интеллектуальных систем в сферах прикладной деятельности.

Методические рекомендации для семинарских (практических) занятий, лабораторного практикума и самостоятельной работы по дисциплине «Алгоритмы искусственного интеллекта на Python» направлены на повышение эффективности освоения знаний, умений, навыков и компетенций, связанных с:

– эволюционным моделированием,
– искусственными нейронными сетями,
– нечёткими системами и др.

Методические рекомендации предлагают указания по всем темам дисциплины Алгоритмы искусственного интеллекта на Python». Методические рекомендации разбиты по темам и содержат набор вопросов для систематизации теоретического материала, полученного на лекционных занятиях, и самостоятельного изучения теории, вопросы (тесты) для текущего контроля на практических занятиях (семинарах), задачи для усвоения практических навыков. Для лабораторного практикума приведены задания, варианты и рекомендации по выполнению лабораторных работ.

Список литературы и информационного обеспечения, приведённый в конце методических указаний, может служить основой для изучения всех рассматриваемых тем. Дополнительная и учебно-методическая литература могут быть использованы обучающимися для закрепления изучаемого материала.

РЕКОМЕНДАЦИИ ПО ОТДЕЛЬНЫМ ТЕМАМ ДИСЦИПЛИНЫ

Тема 2. Основные задачи и методы ИИ

Основные вопросы темы

1. Предыстория, история развития искусственного интеллекта как научного направления.
2. Нейрокибернетика и кибернетика «черного ящика».
3. История развития искусственного интеллекта в России.
4. Основные направления исследований в области искусственного интеллекта.
5. Свойства интеллектуальных информационных систем.

Рекомендации по изучению темы

Вопрос 1 изложен в учебнике [1] на с. 6-9.

Вопрос 2 изложен в учебнике [1] на с. 13.

Вопрос 3 изложен в учебнике [1] на с. 6-9.

Вопрос 4 изложен в учебнике [1] на с. 13-15.

Вопрос 5 изложен в учебнике [1] на с. 22.

Дополнительный материал по вопросам представлен в [2] в разделах 2 и 3, [4] главе 1, 7.

Вопросы для самоподготовки

Рекомендуется после изучения материалов лекций и специальной литературы подготовить ответы на вопросы:

1. Чем сильный ИИ отличается от слабого?
2. Какова основная идея нейрокибернетики?
3. Какова основная идея кибернетики чёрного ящика?
4. Какие задачи решает компьютерная лингвистика?
5. Какие виды анализа выполняют системы при машинном переводе?
6. Какие поколения роботов существуют?
7. Охарактеризуйте японский проект компьютер 5-го поколения?
8. На какие направления исследований делится искусственная жизнь?
9. Приведите примеры компьютерных игр с элементами ИИ и какие методы ИИ в них использовались?
10. Как реализуются творческие задачи в ИИ?

Контрольные тесты

1) Искусственная жизни имеет следующие направления?

Выберите один или несколько ответов:

- a. влажная
- b. твердая
- c. сухая
- d. мягкая
- e. мокрая

2) Какой подход использует Булеву алгебру?

Выберите один ответ:

- a. логический
- b. структурный
- c. имитационный
- d. эволюционный

3) Экспертные знания активно используются в следующих направлениях?

Выберите один или несколько ответов:

- a. нет правильного ответа
- b. экспертные системы
- c. распознавание образов
- d. когнитивное моделирование

4) Сколько поколений роботов существует?

Выберите один ответ:

- a. 1
- b. 2
- c. 4
- d. 3
- e. 5

5) Какое из высказываний определяет основную идею нейрокибернетики?

Выберите один ответ:

a. Не имеет значения, как устроено «мыслящее» устройство. Главное, чтобы на заданные входные воздействия оно реагировало так же, как человеческий мозг.

b. Объект, способный мыслить, — это человеческий мозг. Поэтому любое «мыслящее» устройство должно каким-то образом воспроизводить его структуру.

c. Нет ответа

6) Какие задачи решаются в рамках искусственного интеллекта?

Выберите один или несколько ответов:

a. создание компьютерных игр

b. принятие решений

c. распознавание речи

d. кодирование

e. создание сред разработки информационных систем

7) К системам компьютерной лингвистики относятся:

Выберите один или несколько ответов:

a. нет правильного ответа

b. система распознавания речи

c. машинный перевод

d. система реферирования текстов

e. система генерации музыки

8) Какое из направлений не придает значения тому, как именно моделируются функции мозга?

Выберите один ответ:

a. нейрокибернетика

b. нет правильного ответа

c. кибернетика черного ящика

9) Принцип организации социальных систем используется в направлении?

Выберите один ответ:

a. когнитивное моделирование

b. многоагентные системы

c. нейронные сети

10) Укажите направления искусственного интеллекта (выделить правильные ответы)

Выберите один или несколько ответов:

- a. телекоммуникационные технологии
- b. разработка естественно-языковых интерфейсов и машинный перевод
- c. машинное творчество
- d. технологии открытых систем
- e. обучение и самообучение
- f. распознавание образов
- g. представление знаний и разработка систем, основанных на знаниях
- h. геоинформационные технологии

Тема 2. Обзор библиотек на языке Python, реализующие методы ИИ

Основные вопросы темы

1. Open source библиотеки для задач машинного обучения и ИИ.
2. Языки R и Python.
3. Платформы и облачные сервисы для задачи ИИ.

Рекомендации по изучению темы

Рекомендации см. в методических указаниях для лабораторной работы.

Вопросы для самоподготовки

Рекомендуется после изучения материалов лекций и специальной литературы подготовить ответы на вопросы:

1. Какие библиотеки можно использовать для реализации нейронных сетей?
2. Какая библиотека позволяет выполнять вычисления не на процессоре?
3. Можно использовать в одном проекте библиотеки на языках Python и R?
4. Какие библиотеки можно использовать для обработки естественного языка? Что нужно делать, чтобы они обрабатывали другой (не английский например) язык?

Тема 3. Компьютерное зрение и библиотека OpenCV.

Основные вопросы темы

1. Типы изображений, характеристики изображений.
2. Методы фильтрации изображений
3. Способы преобразования изображения
4. Контуры, края, примитивы
5. Ключевые точки

Рекомендации по изучению темы

Рекомендации см. в методических указаниях для лабораторной работы.

Вопросы для самоподготовки

Рекомендуется после изучения материалов лекций и специальной литературы подготовить ответы на вопросы:

1. Как надо обработать пакет изображений для задачи поиска лиц, задачи выделений краёв, задачи идентификации лица?
2. Какие методы позволяют искать заданные объекты на изображении?
3. Как осуществляется поиск лиц на изображениях?

Тема 4. Рекомендательные системы и библиотека Surprise

Основные вопросы темы

1. Контент-ориентированные рекомендательные системы
2. Коллаборативная фильтрация
3. Метрики персонализации
4. A/B-тестирование

Рекомендации по изучению темы

Рекомендации см. в методических указаниях для лабораторной работы.

Вопросы для самоподготовки

Рекомендуется после изучения материалов лекций и специальной литературы подготовить ответы на вопросы:

1. Какие виды персонализации бывают?
2. Когда можно внедрять в систему персонализацию, какого вида?
3. Как оценить результаты персонализации?

Тема 5. Использование библиотеки Natasha для базовых задач NLP

Основные вопросы темы

1. Токенизация.
2. Морфологический анализ текстов.
3. Сисntаксический анализ текстов.
4. Семантический анализ текстов.
5. Векторизация текстов.
6. Поиск именованных сущностей.
7. Основные методы и классы для решения задач NLP в библиотеке Natasha.

Рекомендации по изучению темы

Рекомендации см. в методических указаниях для лабораторной работы.

Вопросы для самоподготовки

1. Что из себя представляет метод мешок слов?
2. Какие уровни (единицы) токенизации бывают?
3. Чем стемминг отличается от лемматизации?
4. Какие атрибуты формируются при морфологическом анализе у тестовых единиц?

Тема 6. Использование библиотеки PyTorch для реализации искусственных нейронных сетей.

Основные вопросы темы

1. Понятие нейрона.
2. Модель математического нейрона.
3. Персептрон Розенблатта. Правила Хебба.
4. Алгоритм обучения по дельта-правилу.
5. Обучение многослойной нейронной сети методом обратного распространения ошибки.
6. Классификация нейронных сетей.
7. Задачи, решаемые нейронными сетями.
8. Глубинное обучение.
9. Свёрточные нейронные сети.
10. Рекуррентные нейронные сети.

Рекомендации по изучению темы

Вопрос 1 изложен в учебнике [1] на с. 50-53.

Вопрос 2 изложен в учебнике [1] на с. 51.

Вопрос 3 изложен в учебнике [1] на с. 52.

Вопрос 4 изложен в учебнике [1] на с. 55-56.

Вопрос 5 изложен в учебнике [1] на с. 57-58

Вопрос 6 изложен в учебнике [1] на с. 53-55.

Вопрос 7 изложен в учебнике [1] на с. 59-60.

Дополнительный материал по вопросам представлен в [4] главе 4 и в [7].

Вопросы для самоподготовки

Рекомендуется после изучения материалов лекций и специальной литературы подготовить ответы на вопросы:

1. Что такое нейрон?
2. Что такое искусственная нейронная сеть?
3. Какие нейронные сети относятся к глубинным?
4. Что собой представляет процесс обучения?
5. Какие существуют проблемы обучения?
6. Какие достоинства и недостатки у НС?
7. Чем отличается обучение с учителем от обучения без учителя?
8. Какие задачи решают искусственные НС?
9. По каким параметрам можно классифицирования искусственные НС?
10. Какие функции используют в качестве функций активации нейронов?
11. Как инициализируют синаптические веса НС?
12. Что из себя представляет пакетная нормализация?
13. Как выполняется операция свёртка в сверточной НС?
14. Как используется градиент в обучении НС?

Контрольные тесты

1) Искусственные нейронные сети— модели машинного обучения, использующие комбинации распределенных простых операций, зависящих от обучаемых параметров, для обработки входных данных. Какого вида ИНС не существует?

Выберите один или несколько ответов:

☐ а. соревновательные

- [ ] b. прямого распространения
- [ ] c. наивные
- [ ] d. реактивные
- [ ] e. рекуррентные

2) Что из ниже перечисленного относится к персептрону?

Выберите один или несколько ответов:

- [ ] a. нейронная сеть с обратными связями
- [ ] b. многослойная нейронная сеть
- [ ] c. однослойная нейронная сеть
- [ ] d. создан У. Маккалоком и В. Питтом
- [ ] e. создан Ф. Розенблаттом
- [ ] f. нейронная сеть прямого распространени

3) Какие из перечисленных сетей являются рекуррентными?

Выберите один ответ:

- ○ a. нет правильного ответа
- ○ b. персептрон
- ○ c. сеть Хопфилда
- ○ d. сеть радиальных базисных функций

4) Как называется задача, решаемая ИНС и направленая на предсказание значения той или иной непрерывной числовой величины для входных данных?

Выберите один ответ:

- ○ a. Классификация
- ○ b. Кластеризация
- ○ c. Переобучение
- ○ d. Регрессия

5) Какую нейронную сеть обучают с помощью дельта-правила?

Выберите один или несколько ответов:

- [ ] a. сеть Хопфилда

- b. нейронную сеть с обратными связями
- c. однослойную нейронную сеть
- d. нейронную сеть прямого распространения

6) Какую нейронную сеть обучают с помощью алгоритма обратного распространения ошибки?

Выберите один ответ:

- a. многослойную нейронную сеть с обратными связями
- b. нет правильного ответа
- c. многослойную нейронную сеть прямого распространения
- d. Однослойную нейронная сеть

7) Какие задачи не решают нейронные сети?

Выберите один или несколько ответов:

- a. управление
- b. аппроксимация
- c. классификация
- d. память, адресуемая по содержанию
- e. маршрутизация
- f. кодирование

8) Какую функцию не может решить однослойная нейронная сеть?

Выберите один ответ:

- a. логическое «не»
- b. логическое «исключающее или»
- c. логическое «или»

9) Какой из видов машинного обучения основывается на взаимодействии обучаемой системы со средой?

Выберите один ответ:

- a. Обучение с подкреплением
- b. Обучение без учителя
- c. Глубинное обучение

d. Обучение с учителем

10) В какие игры нейросеть еще не научилась обыгрывать человека?

Выберите один ответ:

a. Го

b. Бридж

c. Шахматы

d. «Марио»

Задания для практических занятий и самостоятельной работы

Пример решения

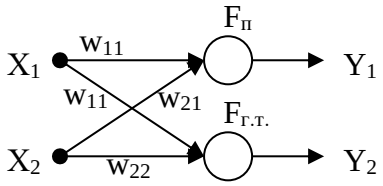
Задача. Просчитать одну итерацию цикла обучения по Δ-правилу однослойной бинарной неоднородной нейронной сети, состоящей из 2 нейронов и имеющей функции активации: гиперболический тангенс (k=1) и пороговую функцию (T=0,7). В качестве обучающей выборки использовать таблицу истинности для операций эквивалентности и дизъюнкции (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.

Описание процесса решения. Для обучения нейронной сети по Δ -правилу необходимо:

1) Графически отобразить структуру нейронной сети. Определить размерность матрицы синаптических весов.
2) Определить обучающую выборку, представив ее в табличном виде.
3) Выбрать входные данные, на которых будет рассматриваться итерация цикла обучения.
4) Следуя алгоритмы обучения по Δ –правилу, просчитать одну итерацию цикла и представить новые синаптические веса в матричном виде.

Решение.

1) По заданию нейронная сеть состоит из двух нейронов, значит, входов у однослойной нейронной сети будет 2 и выходов 2, а синаптических весов 4. Первый нейрон имеет пороговую функцию активации, второй – гиперболический тангенс.



2) По заданию нейронная сеть бинарная, поэтому на ее входы могут подаваться только нули и единицы, так как входа 2, то возможных комбинаций входных значений будет 4 (обучающая выборка будет состоять из 4 векторов). Выход первого нейрона согласно заданию соответствует оператору эквивалентности, а второго – дизъюнкции. Поэтому таблица с обучающей выборкой будет выглядеть следующим образом:

X1	**X2**	**D1**	**D2**
0	0	1	0
0	1	0	1
1	0	0	1
1	1	1	1

3) Пусть в качестве вектора обучения будет рассматриваться 3-ая строка таблицы.

4) Следуя алгоритмы обучения по Δ-правилу выполним 6 шагов

1 шаг: зададим матрицу весов случайным образом из интервала [0,1]:

Wij(1)	**1**	**2**
1	0.7	1
2	0.5	0.2

2 шаг: вектор X={1,0}, вектор D ={0,1}.

3 шаг: вычисление выходных значений нейронной сети (вектор Y).

$T = 0.7;$

$S_1 = x_1 \cdot w_{11} + x_2 \cdot w_{21} = 1 \cdot 0.7 + 0 \cdot 0.5 = 0.7;$

$$Y_1 = \begin{cases} 1, & при\ \ S_1 \geq T \\ 0, & при\ \ S_1 < T \end{cases} = \begin{cases} 1, & при\ \ 0.7 \geq 0.7 \\ 0, & при\ \ 0.7 < 0.7 \end{cases} = 1.$$

$k = 1,$

$S_2 = x_1 \cdot w_{12} + x_2 \cdot w_{22} = 1 \cdot 0.9 + 0 \cdot 0.2 = 0.9,$

$$Y_2 = th\left(\frac{S_2}{k}\right) = \frac{e^{0.9} + e^{-0.9}}{e^{0.9} - e^{-0.9}} \approx 1{,}39.$$

4 шаг:

$\varepsilon_1 = (d_1 - y_1) = (0 - 1) = -1,$

$\varepsilon_2 = (d_2 - y_2) = (1 - 1.39) = -0.39.$

5 шаг: задаем η - коэффициент обучения от 0 до 1 и изменяем веса:

$\eta = 0.8,$

$w_{11}(2) = w_{11}(1) - 0.8 \cdot \varepsilon_1 \cdot x_1 = 0.7 - 0.8 \cdot (-1) \cdot 1 = 1.5,$

$w_{21}(2) = w_{21}(1) - 0.8 \cdot \varepsilon_1 \cdot x_2 = 0.5 - 0.8 \cdot (-1) \cdot 0 = 0.5,$

$\theta_1(2) = \theta_1(1) - 0.8 \cdot \varepsilon_1 = 0.7 - 0.8 \cdot (-1) = 1.5,$

$w_{12}(2) = w_{12}(1) - 0.8 \cdot \varepsilon_2 \cdot x_1 = 0.9 - 0.8 \cdot (-0.39) \cdot 1 = 1.212,$

$w_{22}(2) = w_{22}(1) - 0.8 \cdot \varepsilon_2 \cdot x_2 = 0.2 - 0.8 \cdot (-0.39) \cdot 0 = 0.2.$

Wij(2)	**1**	**2**
1	1.5	1.212
2	0.5	0.2

6 шаг: вычислим среднеквадратичную ошибку (можно выбрать другие методы оценки ошибки)

$$\varepsilon = \sum_{i=1}^{H} (d_i - y_i)^2 = \sum_{i=1}^{2} \varepsilon_i^{\,2} = \varepsilon_1^{\,2} + \varepsilon_2^{\,2} = (-1)^2 + (-0.39)^2 = 1.1521.$$

H- количество нейронов.

Так как мы рассматриваем одну итерацию цикла обучения в любом случае выходим из цикла.

Варианты задач

1. Просчитать одну итерацию цикла обучения по Δ -правилу однослойной бинарной однородной нейронной сети, состоящей из 2 нейронов и имеющей пороговую функцию активации (T=0,7). В качестве обучающей выборки использовать таблицу истинности для операций дизъюнкции и импликации (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.
2. Просчитать одну итерацию цикла обучения по Δ -правилу однослойной бинарной однородной нейронной сети, состоящей из 2 нейронов и имеющей линейную функцию активации (k=0,6). В качестве обучающей выборки использовать таблицу истинности для операций конъюнкции и дизъюнкции (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.
3. Просчитать одну итерацию цикла обучения по Δ -правилу однослойной бинарной однородной нейронной сети, состоящей из 2 нейронов и имеющей сигмоидальную функцию активации (k=1). В качестве обучающей выборки использовать таблицу истинности для операций импликации и конъюнкции (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.
4. Просчитать одну итерацию цикла обучения по Δ -правилу однослойной бинарной однородной нейронной сети, состоящей из 2 нейронов и имеющей функцию

активации гиперболический тангенс (k=1). В качестве обучающей выборки использовать таблицу истинности для операций эквивалентности и импликации (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.

5. Просчитать одну итерацию цикла обучения по Δ -правилу однослойной бинарной неоднородной нейронной сети, состоящей из 2 нейронов и имеющей функции активации: гиперболический тангенс (k=2) и пороговую функцию (T=0,5). В качестве обучающей выборки использовать таблицу истинности для операций эквивалентности и конъюнкции (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.
6. Просчитать одну итерацию цикла обучения по Δ -правилу однослойной бинарной неоднородной нейронной сети, состоящей из 2 нейронов и имеющей функции активации: сигмоидальную (k=1) и линейную (k=0,6). В качестве обучающей выборки использовать таблицу истинности для операций импликации и конъюнкции (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.
7. Просчитать одну итерацию цикла обучения по Δ -правилу однослойной бинарной неоднородной нейронной сети, состоящей из 2 нейронов и имеющей функции активации: линейную (k=0,7) и пороговую (T=0,75). В качестве обучающей выборки использовать таблицу истинности для операций конъюнкции и эквивалентности (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.
8. Просчитать одну итерацию цикла обучения по Δ -правилу однослойной бинарной неоднородной нейронной сети, состоящей из 2 нейронов и имеющей функции активации: пороговую (T=0,8) и сигмоидальную (k=1). В качестве обучающей выборки использовать таблицу истинности для операций конъюнкции и импликации (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.
9. Просчитать одну итерацию цикла обучения по Δ -правилу однослойной бинарной неоднородной нейронной сети, состоящей из 2 нейронов и имеющей функции активации: гиперболический тангенс (k=2) и линейную (k=0,8). В качестве обучающей выборки использовать таблицу истинности для операций дизъюнкции и эквивалентности (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.

10. Просчитать одну итерацию цикла обучения по Δ -правилу однослойной бинарной неоднородной нейронной сети, состоящей из 2 нейронов и имеющей функции активации: гиперболический тангенс (k=2) и сигмоидальную (k=0,9). В качестве обучающей выборки использовать таблицу истинности для операций импликации и дизъюнкции (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.
11. Просчитать одну итерацию цикла обучения по Δ -правилу однослойной бинарной однородной нейронной сети, состоящей из 3 нейронов и имеющей функцию активации гиперболический тангенс (k=3). В качестве обучающей выборки использовать таблицу истинности для X1&X2→X3, X1&X2 и X2→X3 (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.
12. Просчитать одну итерацию цикла обучения по Δ -правилу однослойной бинарной однородной нейронной сети, состоящей из 3 нейронов и имеющей сигмоидальную функцию активации (k=1). В качестве обучающей выборки использовать таблицу истинности для X1→X2&X3, X1&X2 и X1&X3 (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.
13. Просчитать одну итерацию цикла обучения по Δ -правилу однослойной бинарной однородной нейронной сети, состоящей из 3 нейронов и имеющей линейную функцию активации (k=0,9). В качестве обучающей выборки использовать таблицу истинности для X3→X1&X2, X2&X3, X2→X3 (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.
14. Просчитать одну итерацию цикла обучения по Δ -правилу однослойной бинарной однородной нейронной сети, состоящей из 3 нейронов и имеющей пороговую функцию активации (T=0,4). В качестве обучающей выборки использовать таблицу истинности для (X2→X1)&X3 (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.
15. Просчитать одну итерацию цикла обучения по Δ -правилу однослойной аналоговой однородной нейронной сети, состоящей из 3 нейронов и имеющей линейную функцию активации (k=0,9). Синаптические веса и обучающую выборку задать случайным образом (не нули).
16. Просчитать одну итерацию цикла обучения по Δ -правилу однослойной аналоговой однородной нейронной сети, состоящей из 3 нейронов и имеющей сигмоидальную функцию активации (k=0,8). Синаптические веса и обучающую выборку задать случайным образом (не нули).

17. Просчитать одну итерацию цикла обучения по Δ -правилу однослойной аналоговой однородной нейронной сети, состоящей из 3 нейронов и имеющей пороговую функцию активации (T=0,8). Синаптические веса и обучающую выборку задать случайным образом (не нули).
18. Просчитать одну итерацию цикла обучения по Δ -правилу однослойной аналоговой однородной нейронной сети, состоящей из 3 нейронов и имеющей функцию активации – гиперболический тангенс (k=1). Синаптические веса и обучающую выборку задать случайным образом (не нули).
19. Просчитать одну итерацию цикла обучения по Δ -правилу однослойной аналоговой неоднородной нейронной сети, состоящей из 3 нейронов и имеющей функции активации: сигмоидальную (k=1), линейную (k=0,8) и пороговую (T=0,5). Синаптические веса и обучающую выборку задать случайным образом (не нули).
20. Просчитать одну итерацию цикла обучения по Δ -правилу однослойной аналоговой неоднородной нейронной сети, состоящей из 3 нейронов и имеющей функции активации: гиперболический тангенс (k=1), сигмоидальную (k=0,8) и пороговую (T=0,6). Синаптические веса и обучающую выборку задать случайным образом (не нули).

ЛАБОРАТОРНЫЙ ПРАКТИКУМ

Порядок выполнения лабораторных работ может быть произвольным и определяется уровнем освоения компетенций обучающегося.

Тема 3. Компьютерное зрение и библиотека OpenCV

Задание лабораторной работы

Цель работы: получение практических навыков анализа данных на языке Python с использованием библиотеки OpenCV, Face_Recognition.

Задание: используя программу Jupiter Notebook, язык программирования Python, библиотеку OpenCV, Face_Recognition и др. подготовьте набор изображений (с людьми и без) и выполните следующие заданий:

1 Найдите изображения с людьми,

2 Составьте список изображений с указанием, сколько найдено на них людей,

3 Составьте список уникальных лиц и сколько и на каких изображениях они встречались.

Выполните задание с использованием обоих библиотек и сравните результаты. Сделайте выводы .

Отчёт по лабораторной работе должен содержать:

1. Фамилию и номер группы учащегося, задание, вариант.
2. Описание набора данных.
3. Протокол выполнения работы со всеми выводами.
4. Выводы.
5. Код.

Методические указания по выполнению лабораторной работы

Открытая библиотека OpenCV (Open Source Computer Vision Library, https://opencv.org/) содержит алгоритмы для: интерпретации изображений, калибровки камеры по эталону, устранение оптических искажений, определение сходства, анализ перемещения объекта, определение формы объекта и слежение за объектом, 3D-реконструкция, сегментация объекта, распознавание жестов и т.д., т.е. библиотека представляет собой набор типов данных, функций и классов для обработки изображений алгоритмами компьютерного зрения.

Методы обработки изображений и видео, реализованы в разных модулях библиотеки (Таблица 1).

Таблица 1 – Модуль библиотеки OpenCV

Модуль	Описание
opencv_core	ядро: базовые структуры, вычисления (математические функции, генерация псевдослучайных чисел, DFT, DCT, ввод/вывод в XML и т.п.)
opencv_imgproc	обработка изображений (фильтры, преобразования и т. д.)
opencv_highgui	простой UI, загрузка/сохранение изображений и видео
opencv_ml	методы и модели машинного обучения (SVM, деревья принятия решений и т. д.)
opencv_features2d	различные дескрипторы (SURF)
opencv_video	анализ движения и отслеживание объектов (оптический поток, шаблоны движения, устранение фона)
opencv_objdetect	детектирование объектов на изображении (вейвлеты Хаара, HOG и т. д.)
opencv_calib3d	калибровка камеры, поиск стерео-соответстсвия и элементы обработки трехмерных данных
opencv_flann	библиотека быстрого поиска ближайших соседей (FLANN)
opencv_contrib	сопутствующий код, еще не готовый для применения
opencv_legacy	устаревший код, сохраненный ради обратной совместимости
opencv_gpu	ускорение некоторых функций OpenCV за счет CUDA (NVidia)

Полный список модулей бииблиотеки см. https://docs.opencv.org/5.x/.

Основными задачами распознавания изображения с помощью машинного обучения являются (Рисунок 1):

- Захват / загрузка образа (видео / изображения),
- Предобработка (фильтрация и преобразование),
- Извлечение инвариантных признаков / особенностей,
- Обнаружение объектов,
- Анализ и принятие решений.

Для каждой задачи может использоваться свой метод.

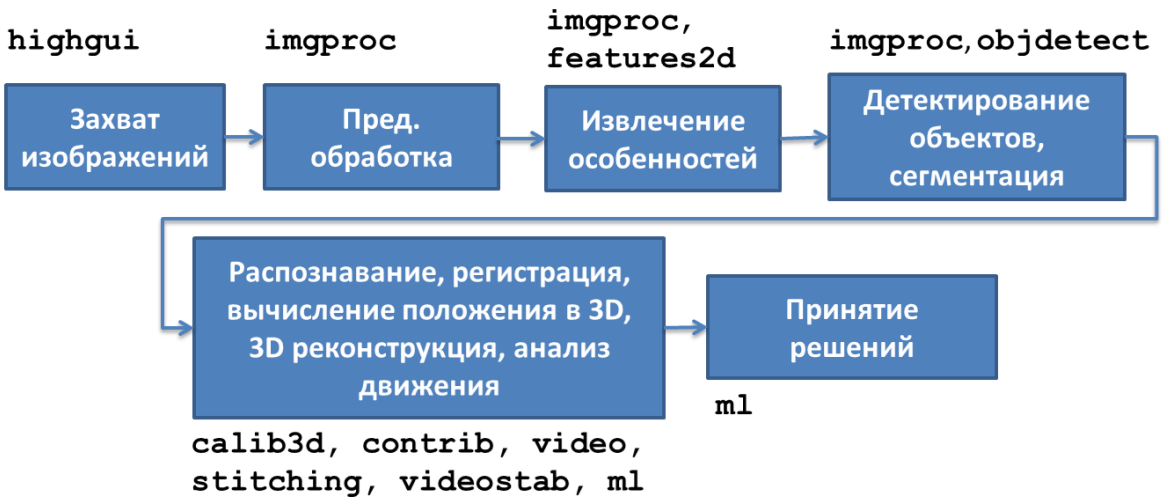


Рисунок 1 – Общая схема работы в библиотеке OpenCV

Популярность библиотеки OpenCV обусловлена еще и тем, что она может использоваться с разными языками, является кроссплатформенной и может использовать аппаратные возможности по ускорению работы своих алгоритмов (Рисунок 2).

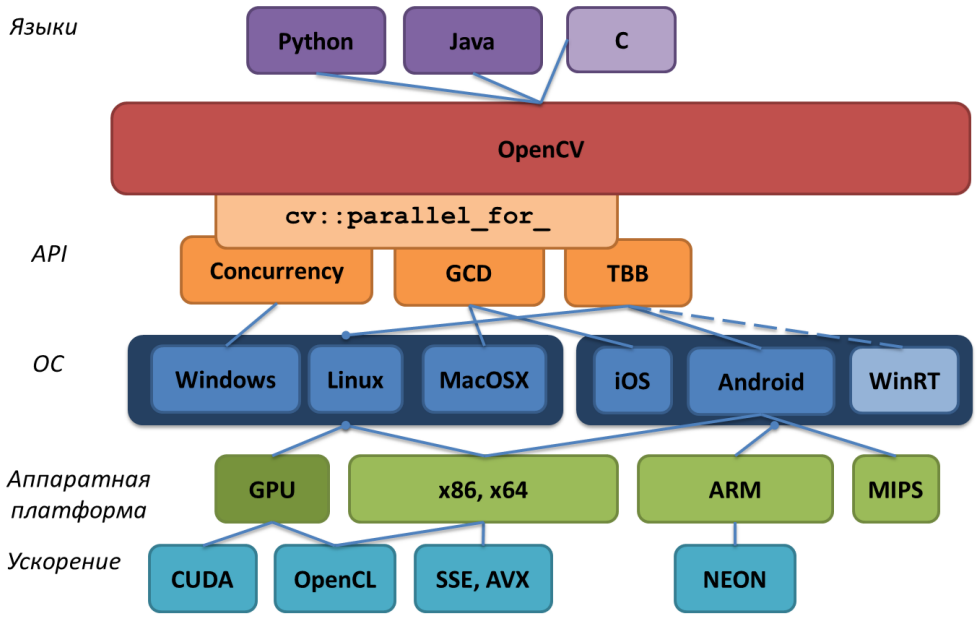


Рисунок 2 – Окружение OpenCV

Установка и импорт библиотеки

Для установки библиотеки воспользуйтесь командой pip.

Пример:

```
# установка библиотеки
!pip install opencv-python
```

Для импорта библиотеки необходимо использовать команду import:

Пример:

```
#Импортировали модуль opencv
import cv2
```

После установки библиотеки можете проверить ее версию.

Пример:

```
# проверяем версию библиотеки
print ("Версия : {0}".format(cv2.__version__))
```

Загрузка и просмотр изображения

Функция загрузки изображений - `cv2.imread()` имеет три режима работы:

- `IMREAD_GRAYSCALE` - преобразует изображение в черно-белое с оттенками серого,
- `IMREAD_UNCHANGED` - загружает изображение без обрезания альфа-канала,
- `IMREAD_COLOR` (по умолчанию) - загружает цветное изображение, используя RGB-каналы.

Функция возвращает массив NumPy (размерность: высота x ширина x 3 для RGB, высота x ширина - для серого).

Пример:

```
# чтение файла с изображением
f1=cv2.imread("1.jpg")

img = cv2.imread("1.jpg", cv2.IMREAD_GRAYSCALE)
```

После загрузки изображения можно проверить его размер (количество пикселей) с помощью атрибута `size` или получить размерность с помощью атрибута `shape`.

Пример:

```
# проверка размера, если 0, то выводим сообщение об ошибке
if img.size == 0:
    sys.exit("Ошибка")

# получение размерности изображения
f1.shape
```

Для просмотра изображения средствами библиотеки OpenCV используется функция imshow().

Параметры функции:

- первый параметр — это строка, которую будет именем окна, в котором откроется изображение,
- второй параметр — это переменная, содержащая загруженное изображение.

Так как изображение будет открыто в новом окне, необходимо добавить функцию ожидания нажатия клавиши, чтобы окно сразу не закрылось, а после этого удалить открытые окна.

Пример:

```
# открытие окна с именем W1 и загруженной картинкой
cv2.imshow("W1", img)
# ожидания нажатия клавиши (чтобы окно сразу не закрылось)
cv2.waitKey(0)

# удаляем все окна
cv2.destroyAllWindows()
```

Сохранение изображений

Для сохранения изображения (измененного или соъхранение в новом файле) используется функция `imwrite()`.

Параметры функции:

- первый параметр – имя файла (полное, с путем),
- второй параметр — это переменная, содержащая загруженное изображение.

Пример:

```
# сохранение изображения в новый файл
cv2.imwrite("2.jpg",img)
```

Если путь не задан, изображение будет сохранено в текущую директорию.

Фильтрация изображений

Под фильтрацией изображений понимают операцию, имеющую своим результатом изображение того же размера, полученное из исходного изображения по некоторым правилам (фильтрам), причем обычно цвет пикселя результирующего изображения обусловлен цветами пикселей, расположенных в некоторой его окрестности в исходном изображении.

Фильтры применяют для того, чтобы обозначить на изображения области интереса или облегчить работу специальным алгоритмам анализа изображения. Основными задачами фильрации являются (Рисунок 3):

- удаление шумов / сглаживание,
- нахождение краёв,
- повышение контраста изображения.

Распространенными типами шума являются:

- «соль и перец» — случайные изолированные черные или белые точки на изображении;
- импульсный — случайные изолированные белые точки на изображении;
- гауссовский — изменении интенсивности по нормальному закону распределения.

Под каждый тип шумов подбирают свой фильтр.

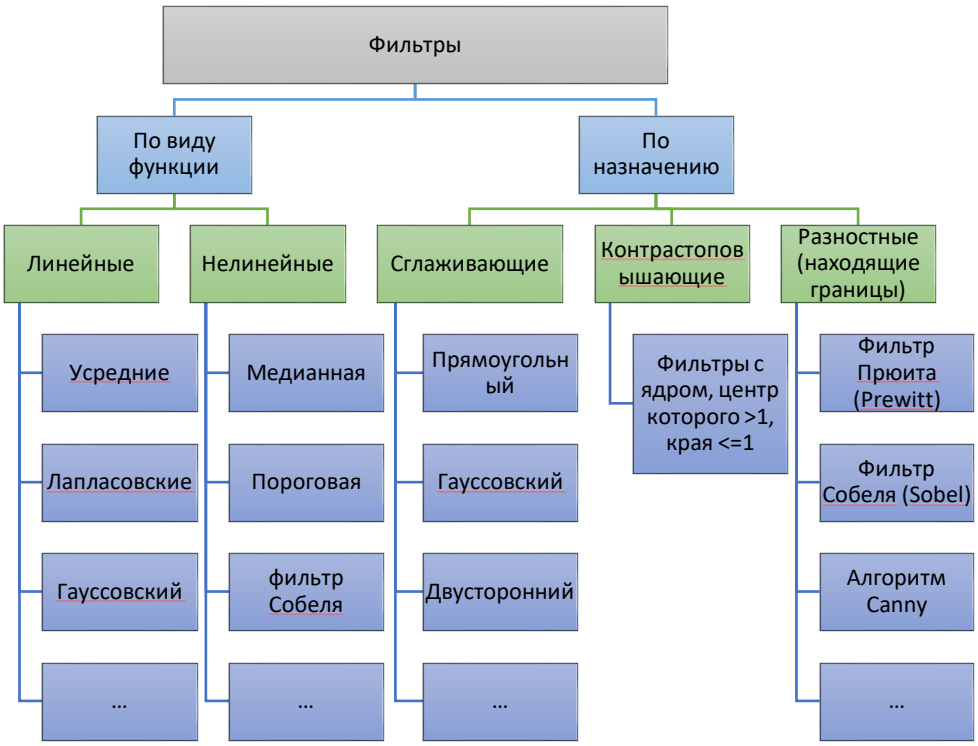


Рисунок 3 – Виды фильтров для обработки изображений

Фильтры можно разделить по типу функции, которую используют для вычисления цвета пиксаля изображения. В матричных фильтрах для вычисления используют маску (фильтр,

матрица свертки, скользящее окно), которая покрывает окрестность точек, влияющих на значение цвета пикселя. Маску задают в виде матрицы NxN с указанием коэффициантов или через функцию, определяющую значение элемента матрицы.
Самые распространенные методы удаления шумов:

- сглаживающие фильтры;
- фильтры Винера;
- медианные фильтры;
- ранжирующие фильтрыэ

Для задачи сглаживания могут использоваться более сложные алгоритмы, которые дают эффект не размытия, а восстановления изображения[1]:

- Edge-avoiding à-trous wavelets (EAW),
- Spatiotemporal Variance-Guided Filtering (SVGF),
- SURE-based Optimization for Adaptive Sampling and Reconstruction (SBF),
- LBF – Machine Learning Approach for Filtering Monte Carlo Noise.

Таблица 2 – Пример фильров библиотеки OpenCV

	Функция и параметры	**Описание**	**Пример**

Медианная фильтрация	`cv2.medianBlur(src,size)`
	`src-input изображение`
	`size - ширина или высота ядра, положительное нечетное число`
		это технология нелинейной обработки сигналов, основанная на статистической теории ранжирования, которая может эффективно подавлять шум. Ее основной принцип заключается в использовании значения точки в цифровом изображении или цифровой последовательности в качестве медианного значения каждой точки в окрестности этой точки.	`img1 = cv2.medianBlur(img,3)`

[1] https://keldysh.ru/papers/2019/prep2019_66.pdf

Двусторонняя фильтрация	cv2.bilateralFilter(src,d,sigmaColor,sigmaSpace) src-input изображение d-Диаметр каждой области пикселя во время фильтрации. Если он отрицательный, программа рассчитает его в соответствии со значением sigmaSpace. sigmaColor - стандартное отклонение цветового пространства [пространство разницы яркости]. Чем больше значение, тем больше цветов объединяется, и, как правило, оно максимально. sigmaSpace - стандартное отклонение координатного пространства, чем больше значение, тем больше значение означает, что до тех пор, пока цвет похож на центральный элемент, будут затронуты более дальние элементы, поэтому чем меньше, тем лучше	это метод нелинейной фильтрации, который сочетает в себе пространственную близость изображения и схожесть значений пикселей, а также учитывает пространственную информацию и схожесть шкалы серого для достижения цели сохранения границ и уменьшения шума.	img4 = cv2.bilateralFilter(img,21,55,55)
Гауссовская фильтрация	cv2.GaussianBlur(src,ksize,sigmaX,[sigmaY])----->dst src-input изображение ksize - размер ядра, положительное нечетное число sigmaX-стандартное отклонение по оси X sigmaY-стандартное отклонение в направлении Y, по умолчанию = sigmaX		img2 = cv2.GaussianBlur(img,(7,7),0)
Фильтр усредне	cv2.blur(src,ksize)---->dst src: выходное изображение ksize: размер ядра, обычно нечетное число		img3 = cv2.blur(img,(7,7))
Фильтр среднего сдвига	cv2.pyrMeanShiftFiltering(src,sp,sr) src-input изображение sp-радиус окна пиксельного пространства, чем больше значение, тем больше теряется детали sr - радиус цветового окна, представляющий диапазон выбранного значения цвета		

https://docs.opencv.org/4.x/d4/d86/group__imgproc__filter.html

Преобразование изображений

Пороговая бинаризациия

Концепция сегментации довольно проста. Как обсуждалось выше в представлении изображения, значениями пикселей могут быть любые значения от 0 до 255. Допустим, мы хотим преобразовать изображение в двоичное изображение, то есть назначить пикселю значение 0 или 1. Для этого мы можем выполнить сегментацию. Например, если значение Threshold(T) равно 125, тогда всем пикселям со значениями больше 125 будет присвоено значение 1, а всем пикселям со значениями, меньшими или равными этому,

будет присвоено значение 0. Давайте сделаем это через код, чтобы получить лучшее понимание.

- cv.THRESH_BINARY
- cv.THRESH_BINARY_INV
- cv.THRESH_TRUNC
- cv.THRESH_TOZERO
- cv.THRESH_TOZERO_INV

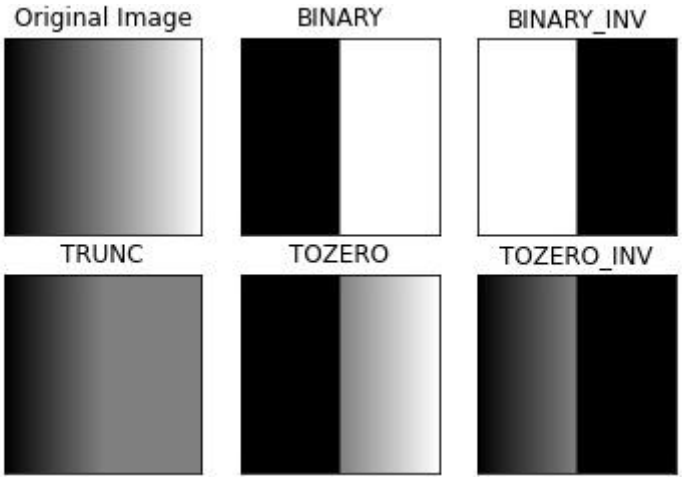


Рисунок 4 – Применение функции Threshold с разными режимами

```
ret, dst = cv2.threshold(src, thresh, maxval, type)
```

- src – Объект класса Mat, представляющий исходное (входное) изображение.
- dst – Объект класса Mat, представляющий целевое (выходное) изображение.
- thresh – переменная типа double, представляющая пороговое значение.
- maxval – переменная типа double, представляющая значение, которое должно быть задано, если значение пикселя превышает пороговое значение.
- тип – переменная целочисленного типа, представляющая тип порога, который будет использоваться.

Функция адаптивного порога бинаризации вычисляет порог соответствующей области в соответствии со значением небольшой области изображения, чтобы получить более подходящее изображение.

```
st = cv2.adaptiveThreshold(src, maxval, thresh_type, type, Block Size, C)
```

- src: входное изображение, можно вводить только одноканальное изображение, обычно изображение в градациях серого
- dst: выходной график
- maxval: когда значение пикселя превышает пороговое значение (или меньше порогового значения, определенного по типу), назначенное значение

- thresh_type: метод расчета пороговых значений, включая следующие 2 типа: cv2.ADAPTIVE_THRESH_MEAN_C; cv2.ADAPTIVE_THRESH_GAUSSIAN_C.
- type: Тип операции бинаризации, такой же, как и функция с фиксированным порогом, включая следующие 5 типов: cv2.THRESH_BINARY; cv2.THRESH_BINARY_INV; cv2.THRESH_TRUNC; cv2.THRESH_TOZERO; cv2.THRESH_TOZERO_INV.
- Размер блока: размер блока на картинке
- C: постоянный член в методе расчета порога

Пример:

```
# бинаризация
seg=cv2.threshold(img, 125, 255, cv2.THRESH_BINARY)

# адаптивная бинаризация
th2 = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_MEAN_C,\
            cv2.THRESH_BINARY,11,2)
th3 = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,\
            cv2.THRESH_BINARY,11,2)
```

Нормализация

Нормализация - это процесс, который изменяет диапазон значения интенсивности пикселей.

Аффинные преобразования

Аффинное преобразование — отображение плоскости или пространства в себя, при котором параллельные прямые переходят в параллельные прямые, пересекающиеся — в пересекающиеся, скрещивающиеся — в скрещивающиеся.

Для простых аффинных преобразований типа изменения размеров любой оси или поворот есть соответствующие функции.

Изменение размеров / растяжения, сжатие

Масштабирование — это не что иное как изменение размеров изображения, его увеличение либо уменьшение. В библиотеке OpenCV для этого существует функция resize. У этой функции, в свою очередь, есть три метода: `INTER_CUBIC`, `INTER_LINEAR` и `INTER_AREA`.

```
# изменение размеров изображения
# второй аргумент новые размеры
# третий аргуметн алгоритм преобразования
img2=cv2.resize(img, (3000,2000), cv2.INTER_AREA)
```

Поворот

```
# возвращает высоту, ширину и каналы
(h, w, d) = img.shape
center = (w // 2, h // 2)
# M — матрица поворота
# первый параметр- координаты точки, вокруг которой будет поворот
# второй параметр -  градусы поворота вокруг центра.
# третий параметр - знак  указывает на направление поворота - по часовой
стрелке,  + против часовой.
# значение параметра на масштаб увеличения, например, 1 - без изменения
масштаба, 2 - увеличение, 0.5 - уменьшение
M = cv2.getRotationMatrix2D(center, 60, -0.5)
img_6 = cv2.warpAffine(img, M, (w, h))
```

Смещение / сдвиг

Произвольное аффинное преобразование

Кроме того, любое аффинное преобразование можно задать через функцию warpAffine.

result = wrapAffine(img,M,dsize)
Описание параметров:
result: вывод изображения
img: входное изображение
M: матрица преобразования, всего шесть параметров, M11, M12, M13, M21, M22, M23
Переписка: out = (M11x+$M12$y+M13,M12x+$M22$y+M23)
аффинное преобразование по приведенной выше формуле
dsize: размер выходного изображения, обычно сначала получаемый методом .shape

Ключом к аффинному преобразованию является получение матрицы преобразования M

Пример:

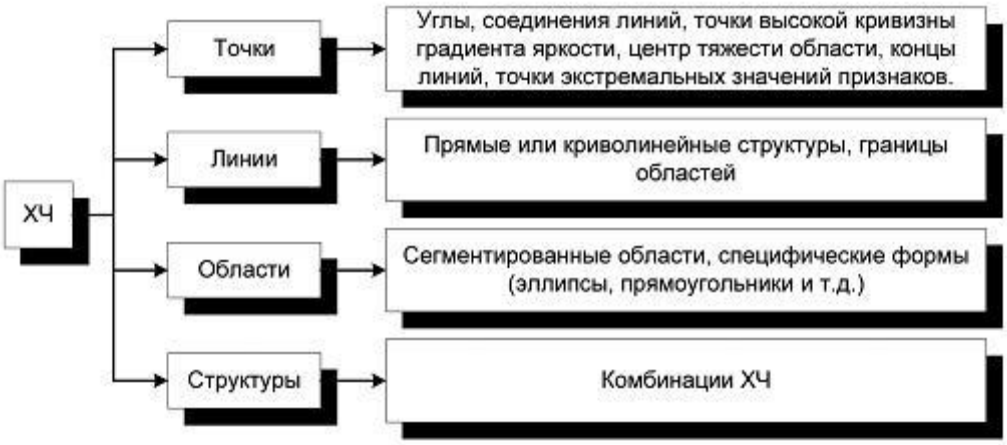
```
import numpy as np

#задание преобразования

T_pos1000 = np.array([
    [1, 0, 1000],
    [0, 1, 1000],
    [0, 0, 1]])

T_rotate = np.array([
    [0, -1, 0],
    [1, 0, 0],
    [0, 0, 1]])

T_scale = np.array([
    [2, 0, 0],
```

```
    [0, 2, 0],
    [0, 0, 1]])
T_neg500 = np.array([
    [1, 0, -500],
    [0, 1, -500],
    [0, 0, 1]])
T = T_pos1000 @ T_rotate @ T_scale @ T_neg500
# Преобразование описания преобразования
T_opencv = np.float32(T.flatten()[:6].reshape(2,3))
# Применение аффинного преобразования
img_transformed = cv2.warpAffine(img, T_opencv, (2000, 2000))
```

Извлечение инвариантных признаков

Это может быть выделение особых точек, выделение контуров или еще каких-либо признаков.

http://wiki.technicalvision.ru/index.php/%D0%92%D1%8B%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%B8_%D0%BE%D0%BF%D0%B8%D1%81%D0%B0%D0%BD%D0%B8%D0%B5_%D1%85%D0%B0%D1%80%D0%B0%D0%BA%D1%82%D0%B5%D1%80%D0%BD%D1%8B%D1%85_%D1%8D%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82%D0%BE%D0%B2_%D0%B8%D0%B7%D0%BE%D0%B1%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F



Характерные черты на изображении имеют следующие виды атрибутов.

1. *Положение:*

- концы отрезка, центр отрезка, центр тяжести области, вершины многоугольников.
- *Геометрические*
- атрибуты: ориентация, длина, кривизна, площадь, периметр, ширина линии, минимальный и максимальный диаметр области, оси симметрии, число и положение особых точек, показатель компактности, и др.
- *Радиометрические*
- атрибуты: контраст, статистика распределения яркости, знак и величина края, автокорреляция.
- *Текстурные*
- атрибуты: матрица смежности, показатель однородности, энергия, энтропия, статистика градиентов текстуры, результаты применения текстурных фильтров, моменты.
- *Топологические*
- атрибуты: связность, соседство, общие точки, пересечение, параллельность, перекрытие, включение.
- *Цветовые/многозональные*
- атрибуты: вектор атрибутов для каждого канала.
- *Динамические*
- атрибуты: атрибуты статических и движущихся объектов.
- *Временные*

8. атрибуты: функции изменения атрибутов со временем.

Выбор конкретных ХЧ и их атрибутов для построения алгоритмов обнаружения должен основываться на следующих основных критериях.

1. *Присутствие/плотность:*

- наличие данных ХЧ на всех используемых изображениях, достаточная плотность ХЧ для покрытия интересующего района.
- *Редкость/Уникальность:*
- редкость конкретной ХЧ на изображении, уникальность ХЧ в окрестности.
- *Инвариантность/Устойчивость:*
- робастность по отношению к геометрическим и радиометрическим искажениям, нечувствительность к шуму.
- *Локализация:*
- возможность точной локализации.
- *Интерпретация:*
- возможность быстрого распознавания и интерпретации.
- *Скорость:* время выделения данного класса ХЧ из исходного изображения.

Выделения контуров

Распознавание объектов производится с помощью цветовой сегментации изображения.

Для этого есть две функции: `cv2.findContours` и `cv2.drawContours`.

В OpenCV для поиска контуров имеется функцией **findContours**

```
findContours( кадр, режим_группировки, метод_упаковки [, контуры[, иерархия[, сдвиг]]])
```

кадр — должным образом подготовленная для анализа картинка. Это должно быть 8-битное изображение. Поиск контуров использует для работы монохромное изображение, так что все пиксели картинки с ненулевым цветом будут интерпретироваться как 1, а все нулевые останутся нулями. На уроке про поиск цветных объектов была точно такая же ситуация.

режим_группировки — один из четырех режимов группировки найденных контуров:

- CV_RETR_LIST — выдаёт все контуры без группировки;
- CV_RETR_EXTERNAL — выдаёт только крайние внешние контуры. Например, если в кадре будет пончик, то функция вернет его внешнюю границу без дырки.
- CV_RETR_CCOMP — группирует контуры в двухуровневую иерархию. На верхнем уровне — внешние контуры объекта. На втором уровне — контуры отверстий, если таковые имеются. Все остальные контуры попадают на верхний уровень.
- CV_RETR_TREE — группирует контуры в многоуровневую иерархию.

метод_упаковки — один из трёх методов упаковки контуров:

- CV_CHAIN_APPROX_NONE — упаковка отсутствует и все контуры хранятся в виде отрезков, состоящих из двух пикселей.
- CV_CHAIN_APPROX_SIMPLE — склеивает все горизонтальные, вертикальные и диагональные контуры.
- CV_CHAIN_APPROX_TC89_L1,CV_CHAIN_APPROX_TC89_KCOS — применяет к контурам метод упаковки (аппроксимации) Teh-Chin.

контуры — список всех найденных контуров, представленных в виде векторов;

иерархия — информация о топологии контуров. Каждый элемент иерархии представляет собой сборку из четырех индексов, которая соответствует контуру[i]:

- иерархия[i][0] — индекс следующего контура на текущем слое;
- иерархия[i][1] — индекс предыдущего контура на текущем слое:
- иерархия[i][2] — индекс первого контура на вложенном слое;
- иерархия[i][3] — индекс родительского контура.

сдвиг — величина смещения точек контура.

Полученные с помощью функции **findContours** контуры хорошо бы каким-то образом нарисовать в кадре. Машине это не нужно, зато нам это поможет лучше понять как выглядят найденные алгоритмом контуры. Поможет в этом ещё одна полезная функция — **drawContours**.

```
drawContours( кадр, контуры, индекс, цвет[, толщина[, тип_линии[, иерархия[,
макс_слой[, сдвиг]]]]])
```

кадр — кадр, поверх которого мы будем отрисовывать контуры;

контуры — те самые контуры, найденные функцией findContours;

индекс — индекс контура, который следует отобразить. -1 — если нужно отобразить все контуры;

цвет — цвет контура;

толщина — толщина линии контура;

тип_линии — тип соединения точек вектора;

иерархия — информация об иерархии контуров;

макс_слой — индекс слоя, который следует отображать. Если параметр равен 0, то будет отображен только выбранный контур. Если параметр равен 1, то отобразится выбранный контур и все его дочерние контуры. Если параметр равен 2, то отобразится выбранный контур, все его дочерние и дочерние дочерних! И так далее.

сдвиг — величина смещения точек контура.

Пример (Рисунок 5)**:**

```
import sys
import numpy as np
# параметры цветового фильтра
hsv_min = np.array((2, 28, 65), np.uint8)
hsv_max = np.array((26, 238, 255), np.uint8)
# меняем цветовую модель с BGR на HSV
hsv = cv2.cvtColor( img, cv2.COLOR_BGR2HSV )
# применяем цветовой фильтр
thresh = cv2.inRange( hsv, hsv_min, hsv_max )
# ищем контуры и складируем их в переменную contours
contours, hierarchy = cv2.findContours( thresh.copy(), cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
# отображаем контуры поверх изображения
cv2.drawContours(img, contours, -1, (255,0,0), 3, cv2.LINE_AA, hierarchy, 1 )
# выводим полученное изображение в окно
cv2.imshow('contours', img) # выводим итоговое изображение в окно
cv2.waitKey()
```

```
cv2.destroyAllWindows()
```

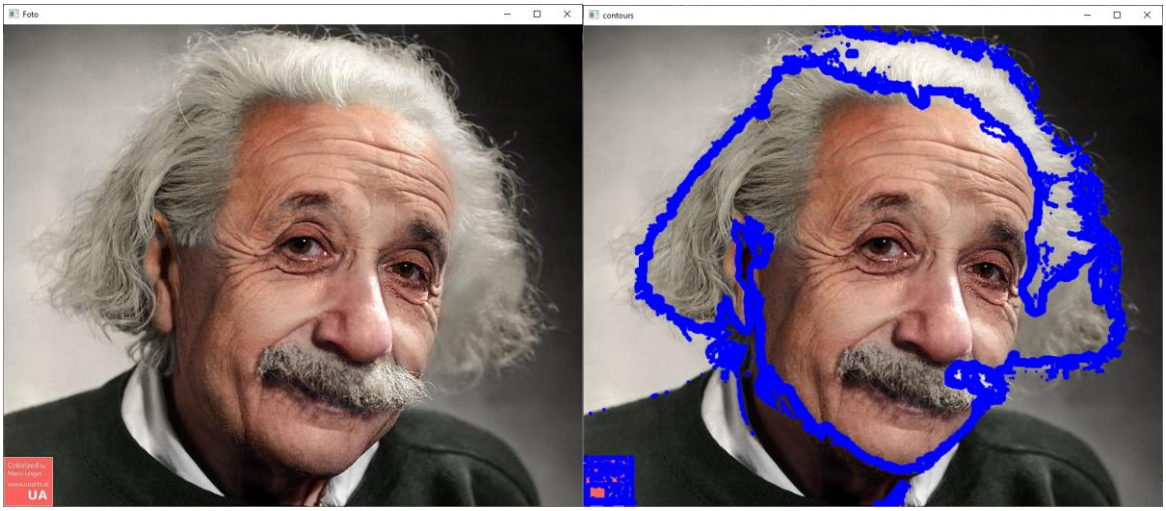


Рисунок 5 – Результат выделения контуров

Выделение краёв

Обнаружение краев - основная проблема в обработке изображений и компьютерном зрении. Цель обнаружения краев - выявить точки с очевидными изменениями яркости цифровых изображений. Значительные изменения атрибутов изображения обычно отражают важные события и изменения атрибутов. К ним относятся: неоднородности по глубине, неоднородности ориентации поверхности, изменения свойств материала и изменения освещения сцены.

Края(границы) — это такие кривые на изображении, вдоль которых происходит резкое изменение яркости или других видов неоднородностей.

Проще говоря, край — это резкий переход/изменение яркости.
Причины возникновения краёв:
* изменение освещенности
* изменение цвета
* изменение глубины сцены (ориентации поверхности)

Получается, что **края отражают важные особенности изображения** и поэтому, целями преобразования изображения в набор кривых являются:
* выделение существенных характеристик изображения
* сокращение объема информации для последующего анализа

- Оператор Кэнни
- Оператор Собеля
- Оператор Лапласа
- Оператор Прюитт
- Оператор Робертса

Самым популярным методом выделения границ является **детектор границ Кенни**.

- edge = cv2. Canny (изображение, порог1, порог2 [, edge [, apertureSize [, L2gradient]]])
- void Canny (изображение InputArray, края OutputArray, double threshold1, double threshold2, int apertureSize = 3, bool L2gradient = false

параметр	**подробности**
образ	Входное изображение
края	Выходное изображение
threshold1	Первый порог для процедуры гистерезиса
порог2	Второй порог для процедуры гистерезиса
размер диафрагмы	Размер диафрагмы для оператора Sobel
L2gradient	Флаг, указывающий, следует ли использовать более точный алгоритм градиента изображения

Алгоритм Canny - это более поздний краевой детектор, разработанный как проблема обработки сигналов. В OpenCV он выводит двоичное изображение, обозначающее обнаруженные края.

https://robocraft.ru/opencv

image — одноканальное изображение для обработки (градации серого)

edges — одноканальное изображение для хранения границ, найденных функцией

threshold1 — порог минимума

threshold2 — порог максимума

aperture_size — размер для оператора Собеля

Пример(Рисунок 6):

```
# обранужение ребер / краев
#Canny Edge - это многошаговый алгоритм
# второй парамерт - порог 1 (Min)
#третий параметр - порог 2 (Max) используется для обнаружения очевидных краев на изображении, но в целом эффект обнаружения не так совершенен.
# Любые края с градиентом интенсивности больше, чем Max, обязательно будут ребрами, меньше Min не ребра.


# выделение краёв
img5 = cv2.Canny(img,50,100)

cv2.imshow("Canny", img5)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

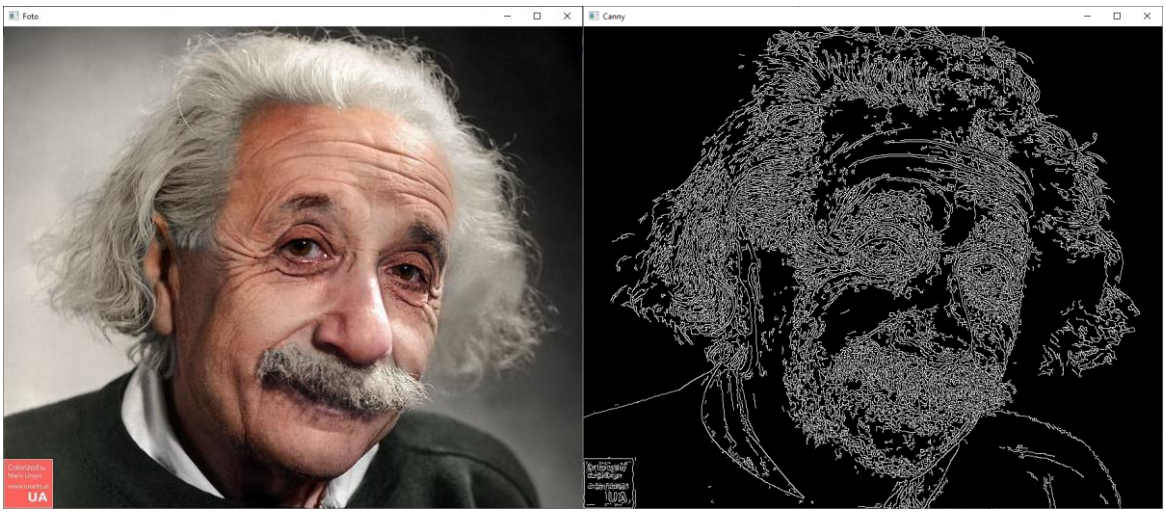


Рисунок 6 – Результат выделения краёв

Поиск геометрических примитивов / Преобразования Хафа

Преобразование Хафа (Hough Transform) — это метод для поиска линий, кругов и других простых форм на изображении.

Преобразование Хафа - это метод преобразования координат из прямоугольной системы координат в полярную систему координат с последующим обнаружением определенных форм (таких как линии и круги) на основе математических выражений.

Эффективность алгоритма в большой степени обусловлена качеством входных данных: границы фигур на этапе предобработки изображения должны быть четко определены. Использование преобразования Хафа на зашумленных изображениях затруднено. Для зашумленных изображений необходим этап предобработки с целью подавления шума.

В OpenCV поддерживаются два различных линейных преобразования Хафа: стандартное преобразование Хафа (SHT, стандартное преобразование Хафа) и преобразование с прогрессивной вероятностью (PPHT, преобразование с прогрессивной вероятностью).

Поиск линий

Функция cv2.HoughLinesP () является вероятностным обнаружением прямой линии.

Примеры:

1) Поиск линий

```
# преобразование в оттенки серого
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
# выделение краёв
edges = cv2.Canny(gray,50,150,apertureSize = 3)
# Преобразование Хафа для поиска кругов
lines =
cv2.HoughLinesP(edges,rho=1,theta=np.pi/180,threshold=100,minLineLength=5,max
LineGap=100)
# отрисовка линий
for x1,y1,x2,y2 in lines[0]:
```

```
    cv2.line(img,(x1,y1),(x2,y2),(0,255,0),2)
```

2) Поиск кругов

```
# преобразование в оттенки серого
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# Преобразование Хафа для поиска кругов
circles1 = cv2.HoughCircles(gray, cv2.HOUGH_GRADIENT, 1, 100, param1=100,
param2=30, minRadius=15, maxRadius=80)
# вывод размерности результата
print (np.shape (circles1))

# отрисовка кругов
circles = circles1[0, :, :]
circles = np.uint16(np.around(circles))
for i in circles[:]:
    cv2.circle(img, (i[0], i[1]), i[2], (0, 255, 0), 3)
    cv2.circle(img, (i[0], i[1]), 2, (255, 0, 255), 10)
```

Выделение ключевых точек

Ключевые точки - характерные точки изображения, т.е. точки, которые сильнее всего отличаются от других областей. Математически ключевая точка является областью с наиболее высоким значением градиента. Например, на шахматной доске характерной точкой будет переход из черной клетки в белую. А у полностью белой доски характерных точек не будет.

Дескрипторы особых точек - алгоритмы, которые дают ключевым точкам математическое описание, т.е. вектор признаков. Дескриптор ключевых точек выполняют поиск ключевых точек, а затем для каждой точки генерирует вектор признаков, например, в KAZE градиент во всех направлениях.

Ключевые точки и их описание используют для сравнения изображений между собой или с шаблонами из обучающей выборки / набора данных.

Таблица 3 – Примеры детекторов ключевых точек библиотеки OpenCV

Детектор	**Описание**	**Пример**
KAZE (open source)		`kaze = cv2.KAZE_create()` `keypoints = kaze.detect(img, None)`
AKAZE (open source)		`akaze = cv2.AKAZE_create()` `keypoints = akaze.detect(img, None)`
BRISK (open source)		`brisk = cv2.BRISK_create()` `keypoints = brisk.detect(img, None)`
Fast (open source)		`fast = cv2.FastFeatureDetector_create()` `keypoints = fast.detect(img, None)`
ORB (open source)	Oriented FAST and Rotated BRIEF– бесплатная, быстрая и эффективная альтернатива алгоритмам обнаружения ключевых точек SIFT и SURF.	`orb = cv2.ORB_create()` `keypoints = orb.detect(img, None)`
SIFT (пропиетарная)	Scale Invariant Feature Transform – алгоритм патентован.	`sift = cv2.SIFT.create()` `keypoints, descriptor = sift.detectAndCompute(gray,`

		None)

Подробнее https://docs.opencv.org/4.5.5/d0/d13/classcv_1_1Feature2D.html.

orb = cv2.ORB_create(, nfeatures, scaleFactor, nlevels, edgeThreshold, firstLevel, WTA_K, scoreType, patchSize, fastThreshold)

- **nfeatures** : Максимальное количество сохраняемых функций, по умолчанию - 500.
- scaleFactor: Коэффициент извлечения пирамиды, больше 1, по умолчанию 1,2. scaleFactor == 2 представляет собой классическую пирамиду, в каждом слое в 4 раза меньше пикселей, чем в предыдущем слое, но такой большой масштабный коэффициент значительно снизит оценку соответствия характеристик. С другой стороны, коэффициент масштабирования, который слишком близок к 1, будет означать, что он покрывает определенный диапазон масштабирования, вам потребуется больше уровней пирамиды, поэтому это повлияет на скорость.
- **nlevels** : Количество уровней пирамиды, по умолчанию 8. Линейный размер минимального уровня будет равен input_image_linear_size / pow (scaleFactor, nlevels-firstLevel).
- **edgeThreshold** : Порог границы, который представляет собой размер границы не обнаруженных объектов, по умолчанию 31. Он должен примерно соответствовать параметру patchSize.
- **firstLevel** : Уровень пирамиды, на котором размещается исходное изображение, по умолчанию - 0. Все предыдущие слои представляют собой увеличенные исходные изображения.
- **WTA_K** : Количество баллов для каждого элемента объектно-ориентированного дескриптора BRIEF. Значение по умолчанию 2 означает, что нужно взять случайную пару точек и сравнить их яркость, чтобы получить ответ 0/1. Другие возможные значения - 3 и 4. Например, это означает, что нам нужны 3 случайные точки (конечно, координаты этих точек случайны, но они генерируют предопределенное начальное число, поэтому краткий дескриптор каждого элемента вычисляет детерминированный прямоугольник пикселей), максимальная яркость найденного победителя и Индекс вывода (0, 1 или 2). Такой вывод будет занимать 2 бита, поэтому для него требуется специальный вариант расстояния Хэмминга, обозначенный как NORM_HAMMING2 (2 бита / бит). Когда WTA_K = 4, мы берем 4 случайные точки для вычисления каждого бина (он также будет занимать 2 бита, которые могут быть 0, 1, 2 или 3).
- **scoreType** : Значение по умолчанию HARRIS_SCORE указывает, что алгоритм Харриса используется для сортировки функций (оценка записывается как KeyPoint :: score, которая используется для сохранения лучших функций); FAST_SCORE - это замещающее значение параметра, значение ключа, генерируемое параметром, немного нестабильно, но вычисляется Будь быстрее.
- **patchSize** : Размер матрицы, используемой объектно-ориентированным дескриптором BRIEF. Значение по умолчанию - 31. Конечно, на меньшем уровне пирамиды воспринимаемая область изображения, покрываемая элементом, будет больше.
- **fastThreshold** : Порог детектора функции FAST, по умолчанию 20

Пример (Рисунок 7)**:**

```
# AKAZE
akaze = cv2.AKAZE_create()
```

```
keypoints = akaze.detect(img, None)
img2 = img.copy()
img2 = cv2.drawKeypoints(img, keypoints, img2, color=(0,255,0))
```

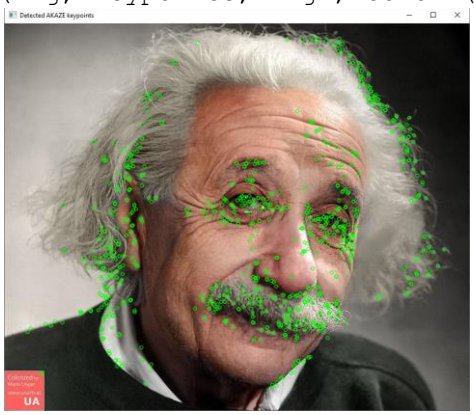


Рисунок 7 - Результат применения алгоритма AKAZE

Пример (Рисунок 8)**:**

```
# SIFT
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
sift = cv2.SIFT.create()
# Получить баллы за особенности
keypoints, descriptor = sift.detectAndCompute(gray, None)
img = cv2.drawKeypoints(image=img1, outImage=img, keypoints=keypoints, flags
= cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS, color=(51, 163, 236))
```

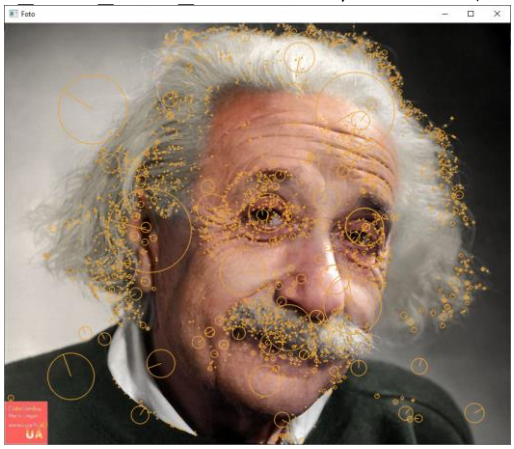


Рисунок 8 – Результат применения алгоритма SIFT

Пример (Рисунок 9)**:**

```
# ORB. Количество установленных ключевых точек невелико, по умолчанию - 500
orb = cv2.ORB_create(nfeatures=200)
# Используйте ORB для обнаружения характерных точек
kp = orb.detect(img, None)
# compute the descriptors with ORB
kp, des = orb.compute(img, kp)
# Нарисовать характерные точки
Img = cv2.drawKeypoints(img, kp, None, color=(0, 255, 0), flags=0)
```

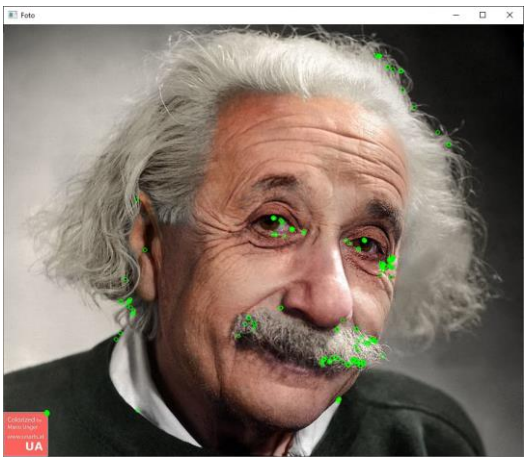


Рисунок 9 – Результат применения алгоритма ORB

Каждый детектор получает свой набор точек, для практических задач можно комбинации детекторов.

Анализ и принятие решений

На этом этапе по найденным признакам на изображения определяться конкретные объекты, и, как правило, их координаты. Так же на этом этапе может происходить сегментация либо какая-то иная высокоуровневая обработка.

Распознавание лиц (Библиотека OpenCV)

`detectMultiScale` — общая функция для распознавания как лиц, так и объектов. Чтобы функция искала именно лица, мы передаём ей соответствующий каскад.

Функция `detectMultiScale` принимает 4 параметра:

1. Обрабатываемое изображение в градации серого.
2. Параметр `scaleFactor`. Некоторые лица могут быть больше других, поскольку находятся ближе, чем остальные. Этот параметр компенсирует перспективу.
3. Алгоритм распознавания использует скользящее окно во время распознавания объектов. Параметр `minNeighbors` определяет количество объектов вокруг лица. То есть чем больше значение этого параметра, тем больше аналогичных объектов необходимо алгоритму, чтобы он определил текущий объект, как лицо. Слишком маленькое значение увеличит количество ложных срабатываний, а слишком большое сделает алгоритм более требовательным.
4. `minSize` — непосредственно размер этих областей.

Признаки Хаара

θ Успешно используются для детектирования лиц, но плохо подходят, например, для детектирования людей в целом.

θ В сочетании с каскадным классификатором позволяют достичь

очень быстрого детектирования.

Основная идея алгоритма заключается в том, что изображение может быть описано распределением градиентов интенсивности или направления краев. Как правило, построение этих дескрипторов происходит путем разбиения изображения на ячейки, и присвоения каждой ячейке гистограммы направлений градиентов для пикселей внутри ячейки, их комбинация и является дескриптором. С целью увеличения точности обрабатываемое изображение, как правило, делают чёрно-белым, а локальные гистограммы нормализуют по контрасту относительно меры интенсивности, вычисляемой на большем фрагменте изображения. Нормализация по контрасту позволяет добиться большей инвариантности дескрипторов к освещению.

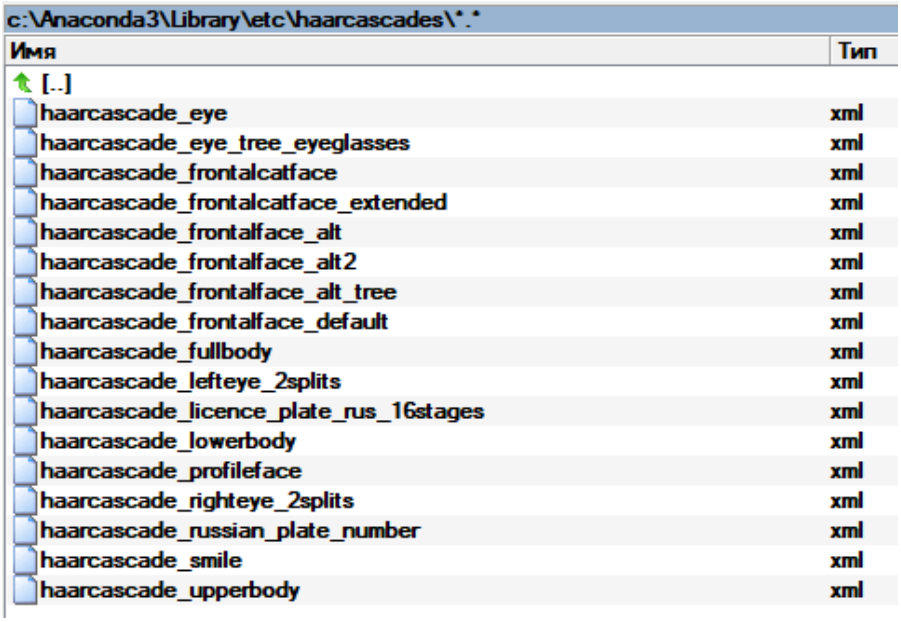


Рисунок 10 – Устанавливаемые с библиотекой каскады HAAR для обнаружения различных объектов

Пример (Рисунок 11)**:**

```
face_cascade =
cv2.CascadeClassifier('C:\Anaconda3\Library\etc\haarcascades\haarcascade_fron
talface_alt2.xml')

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faces = face_cascade.detectMultiScale(
    gray,
    scaleFactor= 1.1,
    minNeighbors= 5,
    minSize=(10, 10)
)
faces_detected = "Лиц обнаружено: " + format(len(faces))
print(faces_detected)
# Рисуем квадраты вокруг лиц
for (x, y, w, h) in faces:
    cv2.rectangle(img, (x, y), (x+w, y+h), (255, 255, 0), 2)
```

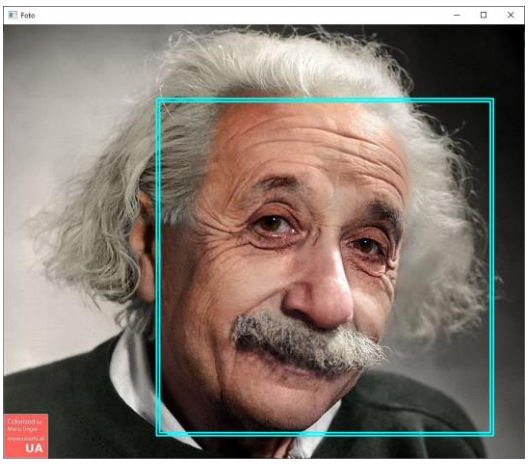


Рисунок 11 – Результат применения каскада HAAR для обнаружения лица

Пример работы с библиотекой

1) Пример работы с изображением в OpenCV

```
# установка библиотеки
!pip install opencv-python


# импорт библиотеки
import cv2

# проверяем версию библиотеки
print ("Версия : {0}".format(cv2.__version__))

# чтение файла с изображением
f1=cv2.imread("1.jpg")

# получение размеров изображения
f1.shape

# открытие изображения в отдельном окне (если файл не в той же папке, выходит
ошибка) в сером цвете
# второй необязательный параметр - цветовое пространство
# RGB — cv2.IMREAD_COLOR
# оттенки серого — cv2.IMREAD_GRAYSCALE
# функция возвращает массив NumPy (высота x ширина x 3 для RGB,  высота x
ширина - для серого).
img = cv2.imread("1.jpg", cv2.IMREAD_GRAYSCALE)
# проверка размера, если 0, то выводим сообщение об ошибке
if img.size == 0:
    sys.exit("Ошибка")
# открытие окна с именем W1 и загруженной картинкой
cv2.imshow("W1", img)
# ожидания нажатия клавиши (чтобы окно сразу не закрылось)
cv2.waitKey(0)

# удаляем все окна
cv2.destroyAllWindows()

# сохранение изображения в новый файл
cv2.imwrite("2.jpg",img)

# обращение к конкретному пикселю
# OpenCV хранит каналы формата RGB в обратном порядке (синий, зеленый и
красный)
img[0, 0]
```

```
# изменение значения пикселя
img[55,55] = 255

# изменение размеров изображения
# второй аргумент новые размеры
# третий аргуметн алгоритм преобразования
img2=cv2.resize(img, (3000,2000), cv2.INTER_AREA)

# вырезаем фрагмент из изображения (прямоугольный по координатам)
img3=img[100:500, 100:500]

# рисование графических примитивов на изображении
# линия
img4=cv2.line(img,(0,0),(150,150),(255,255,255),15)
# прямоугольник
img4=cv2.rectangle(img,(15,25),(200,150),(0,0,255),15)
# круг
img4=cv2.circle(img,(100,63), 55, (0,255,0), -1)

# ломаная линия
import numpy as np
pts = np.array([[10,5],[20,30],[70,20],[50,10]], np.int32)
img4=cv2.polylines(img, [pts], True, (0,255,255), 3)

# надпись
font = cv2.FONT_HERSHEY_SIMPLEX
img4=cv2.putText(img,'Hello, world',(0,130), font, 1, (200,255,155), 2,
cv2.LINE_AA)

# обранужение ребер / краев
#Canny Edge - это многошаговый алгоритм
# второй парамерт - порог 1 (Min)
#третий параметр - порог 2 (Max) используется для обнаружения очевидных краев
на изображении, но в целом эффект обнаружения не так совершенен.
# Любые края с градиентом интенсивности больше, чем Max, обязательно будут
ребрами, меньше Min не ребра.
img5 = cv2.Canny(f1,50,100)

cv2.imshow("W1", img5)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

2) Пример аффинных преобразований[2]

```
import numpy as np
from cv2 import cv2
lena = cv2.imread('black.bmp')
# Первая строка и затем столбец, сколько строк соответствует высоте
# Сколько столбцов соответствует ширине
width,height = lena.shape[:2]
# Dx и dy должны быть целыми числами
lena_1 = cv2.resize(lena,dsize=(int(width*2),int(height*2)))
# Горизонтальное отражение, положительное число по горизонтали, отрицательное
число по вертикали, 0 по горизонтали плюс вертикаль
lena_2 = cv2.flip(lena_1,flipCode=1)
# Создать массив переводов
#1*x+0*y+50,0*x+1*y+50=x+50,y+50
# Сдвиг 50 влево вниз
```

[2] https://russianblogs.com/article/7364707431/

```
m_move = np.float32([[1,0,50],[0,1,50]])
# Использование матрицы перевода для аффинного преобразования
lena_3 = cv2.warpAffine(lena_2,m_move,dsize=(width,height))
# Создать матрицу вращения
m_ratation = cv2.getRotationMatrix2D((width/2,height/2),45,1)
# Использование матрицы вращения для аффинного преобразования
lena_4 = cv2.warpAffine(lena_3,m_ratation,dsize = (width,height))
# point_0 - три точки исходного изображения
# точка_1 - это три точки новой картинки, эти три точки соответствуют друг
другу
points_0 = np.float32([[0,0],[0,height-1],[width-1,0]])
points_1 = np.float32([[0,50],[50,90],[200,30]])
# Используйте это однозначное соответствие для генерации матрицы
m_complex = cv2.getAffineTransform(points_0,points_1)
#rows - количество строк, cols - количество столбцов.
rows,cols= lena_4.shape[:2]
lena_5 = cv2.warpAffine(lena_4,m_complex,dsize=(cols,rows))
cv2.imshow('before',lena)
cv2.imshow('lena_1',lena_1)
cv2.imshow('lena_2',lena_2)
cv2.imshow('lena_3',lena_3)
cv2.imshow('lena_4',lena_4)
cv2.imshow('lena_5',lena_5)
cv2.waitKey()
cv2.destroyAllWindows()
```

3) Пример распознавания лиц (библиотека OpenCV)

```
# импорт библиотеки
import cv2
import matplotlib.pyplot as plt

# загрузка файла
img = cv2.imread('D:/3.jpg', cv2.IMREAD_COLOR)
# переводим в серые тона
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)


# вывод изображения
plt.imshow(img)
plt.imshow(gray_img, cmap='gray')

#  Используем каскадный классификатор для лица HAAR
# соответсвующий путь к файлу необходимо найти на машине (при указании
другого пути для файла выходит ошибка)
#  каскады можно найти по адресу
https://github.com/opencv/opencv/tree/master/data/haarcascades
fc
=cv2.CascadeClassifier('C:\Anaconda3\Library\etc\haarcascades\haarcascade_fro
ntalface_alt2.xml')
# обнаружение всех лиц на изображении
faces = fc.detectMultiScale(gray_img)
print('Количество лиц: ', len(faces))

# рисуем рамки вокруг найденных лиц
for (x, y, w, h) in faces:
    cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)
plt.imshow(img)
```

Распознавание лиц (Библиотека Face_Recognition)

Распознавание лиц - это идентификация человека по его лицу, например, для реализации механизма аутентификации (разблокировка смартфона).

Существует множество методов решения этой задачи SIFT, LBP, HOG дескрипторов, глубокие нейронных сетей, например FaceNet, DeepFace, OpenFace, Deep Residual Learning for Image Recognition.

Для задачи распознавания лица нужно выполнить подзадачи:

- обнаружение лица на изображении,
- кодирование данных о лице (представление лица в виде 128-мерного вектора признаков),
- поиск черт лиц,
- сопоставление лица с шаблоном (с известными лицами).

Face_Recognition (https://github.com/ageitgey/face_recognition) - *библиотека* распознавания лиц с открытым исходным кодом для Python, включает в себя функции распознавания лиц библиотеки dlib и является надстройкой над ней.

Библиотека dlib (https://github.com/davisking/dlib, http://dlib.net/) содержит реализацию алгоритмов машинного обучения (несколько реализаций SVM, несколько реализаций нейронных сетей, подборку функций матричной геометрии, подборку алгоритмов градиентного спуска, SURF|HOG). Для поиска лиц dlib использует обученный HOG-каскад.

Библиотека OpenCV может использоваться для загрузки и обработки изображений, а dlib/ Face_Recognition — для детекции лиц.

С документацией дидлиотеки можно ознакомиться по адресу https://face-recognition.readthedocs.io/en/latest/index.html.

Установка и импорт библиотеки

Вначале надо учтановить dlib из оболочки или через Anaconda:

```
conda install -c conda-forge dlib.
```

Далее установить из блокнота face_recognition.

```
!pip install face_recognition
```

Для импорта библиотеки необходимо использовмать команду import:

```
import face_recognition as fr
```

Загрузка изображений

Для загрузки изображений библиотека Face_Recognition предоставляет функцию *load_image_file load_image_file(file, mode='режим'),* которая загружает изображение в массив библиотеки Numpy.

Параметры функции:

– первый параметр (file): имя загружаемого файла изображения,
– mode: формат изображения: «RGB» (по умолчанию) или «L».

Пример (Рисунок 12**):**

```
image1 = fr.load_image_file('D:/5.jpg')
image2= fr.load_image_file('D:/5.jpg', mode='L')
```

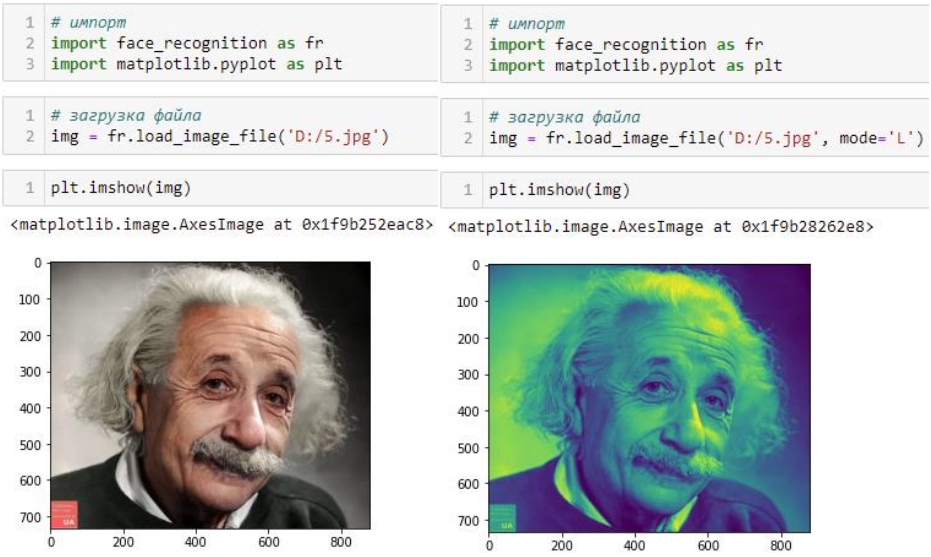


Рисунок 12 – Пример загрузки изображения

Кодировака изображений

После загрузки изображения его данные можно просмотреть с помощью функции print (Рисунок 13) и увидеть массив, соответсвующий размерности изображению и выбранному формату (RGB – 3 канала).

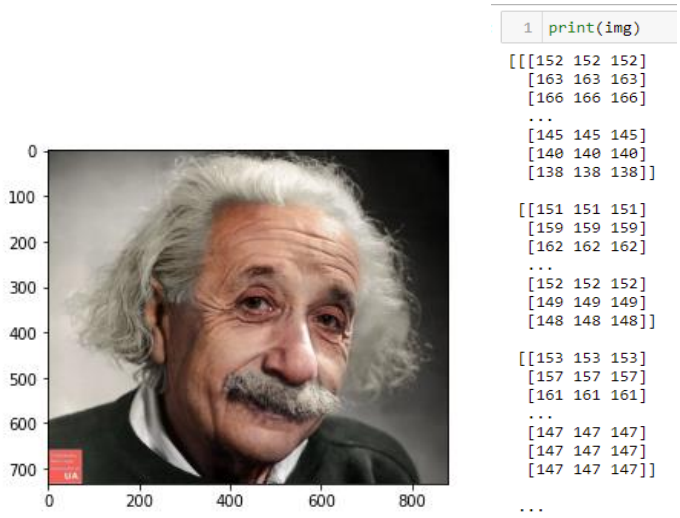


Рисунок 13 – Загруженное изображение

Для задачи детектирования (распознавания) лиц используется кодированное представление – вектор признаков (размерность может быть 128, 256 или 512).

Данный вектор формируется алгоритмом и признаки не могут быть интерпретировы для использования человеком (алгоритм сам находит эти признаки).

Вектора для одних и тех же лиц одинаковы, если лица разные, то угол между векторами не будет равен нулю. Это означает, что для определения степени сходства двух лиц достаточно измерить угол между их векторами.

В библиотеке Face_Recognition для получения вектора признаков используется функция *face_encodings(face_image, known_face_locations=None, num_jitters=N)*, которая возвращает 128-мерный вектор признаков для каждого лица на изображении.

Параметры функции *face_encodings*:

- face_image: загруженное изображение,
- known_face_locations: необязательный параметр, координаты границ лиц, если они известны,
- num_jitters = N (по умолчанию 1): количество повторных выборок при вычислении кодировки, чем выше, тем точнее, но медленнее скорость.

Пример (Рисунок 14):

```
img_code=fr.face_encodings(img)
```

```
1 img_code=fr.face_encodings(img)
```

```
1 print(img_code)
```

```
[array([-0.08187886,  0.03808367,  0.03709373, -0.03347099, -0.0070551 ,
       -0.03495234, -0.06054493, -0.04017232,  0.17318228, -0.10732149,
        0.11848658,  0.06458141, -0.21167736, -0.04198301,  0.01921765,
        0.14788547, -0.14215797, -0.04268409, -0.14635117, -0.10254101,
       -0.04390452,  0.08646965,  0.08363062, -0.10068246, -0.02939744,
       -0.36324966, -0.06823933, -0.04722461,  0.05894782, -0.05593463,
        0.03356535,  0.02545134, -0.19075812, -0.05794752,  0.03204786,
        0.03466977, -0.03555492, -0.02160053,  0.20194075,  0.13313827,
       -0.19317801,  0.03423087,  0.03460854,  0.23543295,  0.23946439,
       -0.00974654,  0.03743351, -0.07905183,  0.0844241 , -0.21292186,
        0.09202813,  0.08079303,  0.19094168,  0.05125459,  0.13409927,
       -0.12332872,  0.00661359,  0.10565154, -0.15567994,  0.08122978,
        0.08973004, -0.05660111, -0.02581698,  0.02679678,  0.12942539,
        0.06207754, -0.11191515, -0.13902341,  0.10208006, -0.11698095,
        0.00123641,  0.0587865 , -0.0683361 , -0.16475929, -0.31880921,
        0.03664724,  0.34849843,  0.14607979, -0.13187025,  0.01177806,
       -0.0856711 , -0.0903268 ,  0.01765568,  0.14481696, -0.08078516,
       -0.02435208,  0.03297128,  0.0264859 ,  0.12268842,  0.03102197,
       -0.10568522,  0.16484627, -0.0411277 ,  0.02444832, -0.06262729,
        0.03141201, -0.12057207, -0.04686332, -0.06948614, -0.10592335,
       -0.00156569, -0.12329355,  0.04514425,  0.10720998, -0.21470238,
        0.19085449, -0.03112776, -0.01748933,  0.05999868,  0.08090753,
        0.00471095,  0.01419915,  0.09785823, -0.22577029,  0.21540163,
        0.23496424, -0.02004887,  0.13652945,  0.05786948,  0.06973504,
        0.01640278,  0.05938628, -0.12387886, -0.09611931, -0.09565987,
        0.04622603,  0.07829622,  0.06930646])]
```

Рисунок 14 – Результат кодирования изображения

Поиск лиц на изображении

Гистограммы ориентации градиентов (гистограммы ориентированных градиентов, Histograms of Oriented Gradients, HOG) —признаковое описание (Рисунок 15), характеризующее форму объекта, изначально предложенное для детектирования пешеходов (основан на подсчете количества направлений градиента в локальных областях изображения).

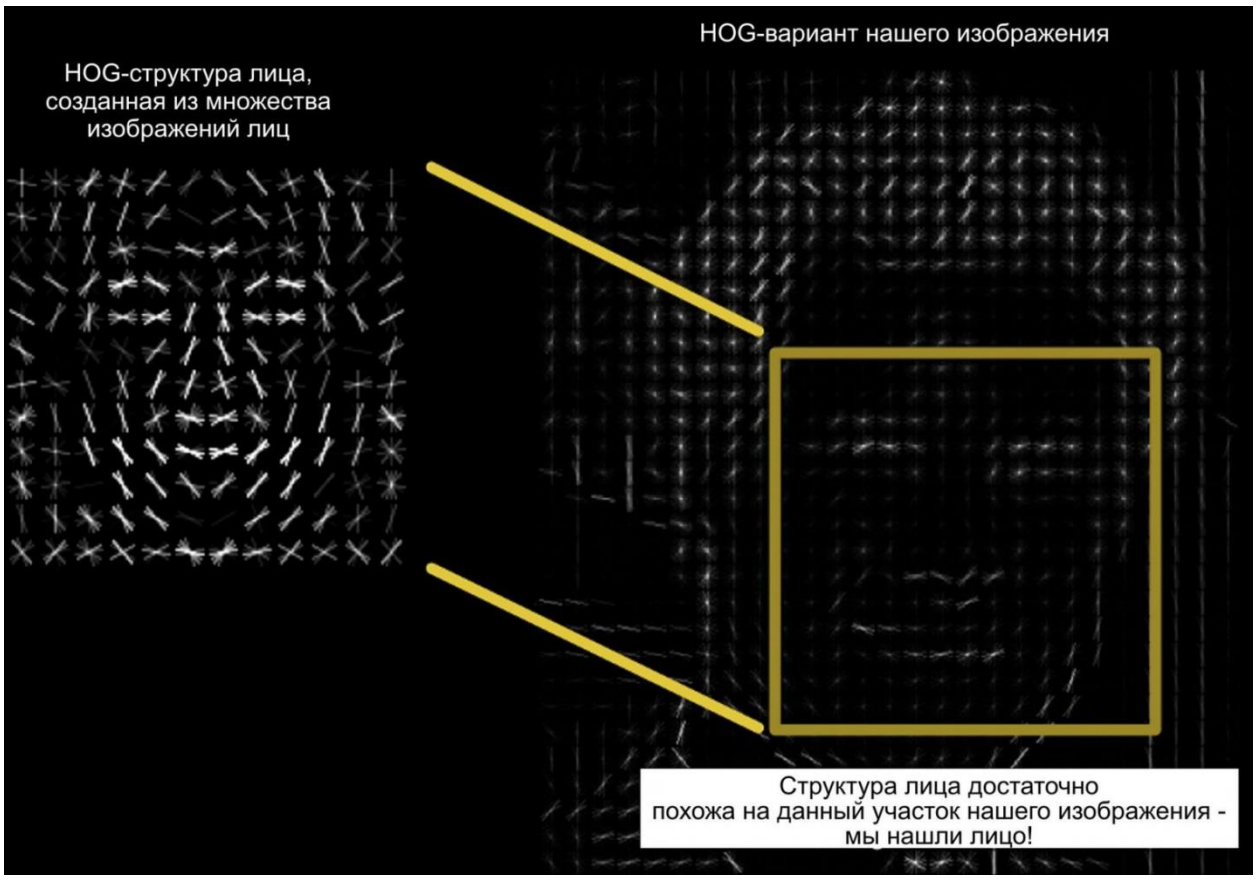


Рисунок 15 – Иллюстация метода HOG[3]

Сверточная нейронная сеть (convolutional neural network, CNN) - специальная архитектура искусственных нейронных сетей (Рисунок 16), предложенная Яном Лекуном в 1988 году и нацеленная на эффективное распознавание образов, входит в состав технологий глубокого обучения.

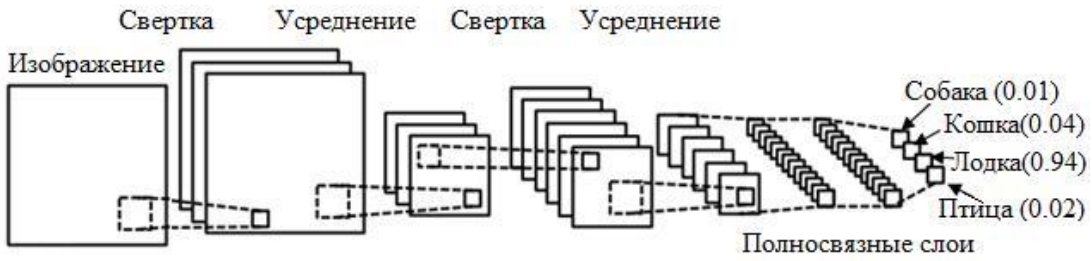


Рисунок 16 – Общая слема слоёв CNN

Функция *face_locations(face_image,number_of_times_to_upsample=N,model="имя модели поиска лиц")* получает местоположение лица, а возвращаемая структура представляет собой список. Каждое лицо сохраняется как кортеж, представляющий координаты верхнего левого и нижнего правого углов прямоугольника, обрамляющего лицо (x1, y1, x2, y2 - вверху, справа, внизу, слева). Если лиц несколько, то возвращается кортеж кортежей.

Параметры:

[3] https://habr.com/ru/post/306568/

– face_image: загруженное изображение,
– number_of_times_to_upsample =N (1 по умолчанию): сколько раз найти лицо по образцу изображения,
– model = "hog" (по умолчанию) /«cnn»: какую модель распознавания лиц использовать. Точность «hog» невысока, но он работает быстрее на ЦП, а «cnn» более точен и глубже (лучше использовать ускорение GPU / CUDA с этим методом), полученный координаты с помощью разных моделей могут отличаться (Рисунок 17).

Пример:

```
# поиск лиц на изображении
face_loc = fr.face_locations(img)
# получаем количество лиц
print(len(face_loc))
# получаем координаты лиц
print(face_loc)
# поиск лиц на изображении с помощью модели CNN
face_loc1 = fr.face_locations(img, model='cnn')
```

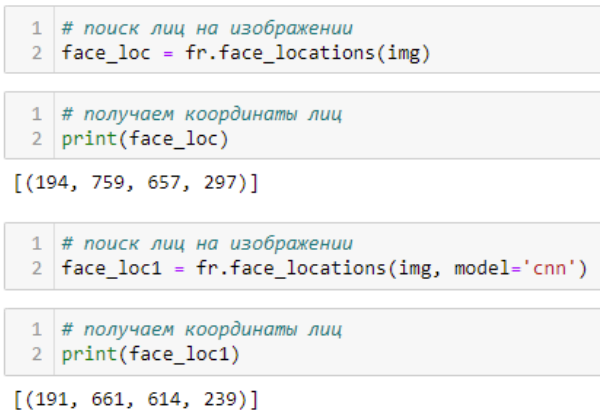


Рисунок 17 – Пример получения координат разными моделями

Получение координат элементов (черт) лиц (антропометрических точек)

Выделяется 68 специфических антропометрических точек (*меток*), имеющихся на каждом лице, — выступающая часть подбородка, внешний край каждого глаза, внутренний край каждой брови и т.п.

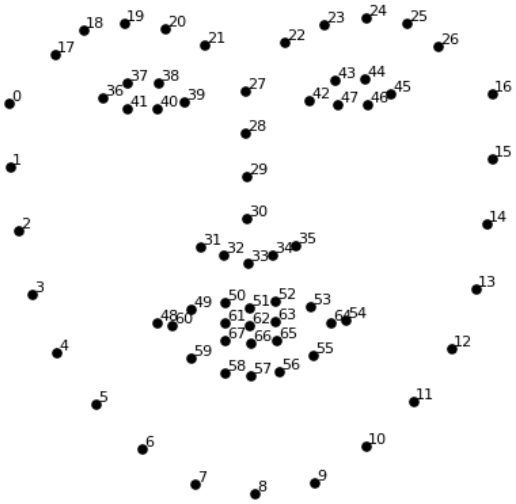


Рисунок 18 – Антропоморфные точки (метки)

Расстановка меток на изображении лица не является самой целью, это всего лишь инструмент для решения других задач: отслеживание двигательных единиц, участвующих в формировании эмоций, направление взгляда, определения эмоционального состояния человека, верификация и идентификация человека, для базовых преобразований (вращение и масштабирование, которые сохраняют параллельность линий) и наложения масок.

Для поиска антропометрических точек в библиотеке Face_Recognition используется функция *face_landmarks(face_image,face_locations=None,model="large")*.

Параметры:

- face_image: загруженное изображение,
- face_locations = None: необязательный параметр, значение по умолчанию - None, которое представляет каждое лицо в декодированном изображении. Если вы введете face_locations()[i], вы можете указать лицо для декодирования,
- model = "large" (по умолчанию) / "small": модель выходного объекта. Если выбран вариант «small», извлекаются только три черты лица: левый глаз, правый глаз и кончик носа.

Функция возвращает список словарей List [Dict [str, List [Tuple [Any, Any]]]], ключ - это черта лица (например, нос, левый_глаз и т.), значение - это список, состоящий из положений ключевых точек черты лица, а положение ключевой точки - это кортеж (см. Рисунок 19).

Пример:

```
# получение антропометрических точек
face_landmarks_list = fr.face_landmarks(img)
```

```
# получение антропометрических точек
face_landmarks_list = fr.face_landmarks(img)
```

```
print(face_landmarks_list)
```

```
[{'chin': [(266, 332), (257, 392), (248, 453), (250, 515), (273, 567), (315, 610), (364, 641), (416, 670), (469, 684), (512, 68
3), (541, 656), (570, 627), (596, 593), (622, 556), (642, 512), (651, 469), (653, 428)], 'left_eyebrow': [(376, 286), (406, 26
1), (448, 247), (490, 256), (525, 285)], 'right_eyebrow': [(593, 304), (619, 295), (649, 299), (669, 322), (670, 356)], 'nose_b
ridge': [(555, 359), (559, 400), (564, 440), (568, 481)], 'nose_tip': [(488, 491), (512, 506), (537, 520), (559, 516), (578, 50
6)], 'left_eye': [(418, 338), (442, 326), (469, 331), (486, 357), (462, 359), (436, 352)], 'right_eye': [(573, 381), (595, 36
6), (620, 370), (632, 393), (616, 398), (592, 392)], 'top_lip': [(412, 543), (464, 547), (506, 554), (526, 563), (544, 563), (5
60, 572), (568, 588), (559, 586), (537, 576), (519, 574), (498, 566), (426, 549)], 'bottom_lip': [(568, 588), (544, 595), (522,
597), (503, 592), (481, 584), (449, 568), (412, 543), (426, 549), (492, 567), (514, 575), (531, 579), (559, 586)]}]
```

Рисунок 19 – Вывод антпропометрических точек

Сравнение лиц на изображении

Для измерения сходства лиц используются вектора признаков, т.е. сравниваются не изображения, а их вектора.

В функции *compare_faces(known_face_encodings, face_encoding_to_check, tolerance=0.6)* для измерения меры сходства используют нутреннее произведение двух векторов и некое пороговое значение (если расстояние между двумя векторами признаков лица находится в пределах диапазона пороговых значений, они считаются одним и тем же человеком), определяющее меру сходства.

Параметры:

- known_face_encodings: список известных кодировок лиц,
- face_encoding_to_check: данные кодировки одного лица для сравнения
- tolerance (пороговое значение = 0,6 (по умолчанию): лица считаются соврадающими, если расстояние между двумя векторами соответсвует порогу. Чем меньше значение, тем строже сравнение.

Функция возвращает список значений True или False, указывающих результаты сопоставления для каждого члена списка known_face_encodings.

Пример:

```
# загружаем 2 фото
known_image = fr.load_image_file("D:/01.jpg")
unknown_image = fr.load_image_file("D:/02.jpg")
# кодируем фото
known_encoding = fr.face_encodings(known_image)[0]
unknown_encoding = fr.face_encodings(unknown_image)[0]
# сравниваем
results = fr.compare_faces([known_encoding], unknown_encoding)
print(results)
```

Для сравнения векторов можно использовать евклидово расстояние (чем меньше, тем больше похожи вектора). Для вычисления евклидово расстояния используется функция *face_distance(face_encodings,face_to_compare).*

Параметры:

- face_encodings: список кодировок лиц,
- face_to_compare: кодировка для сравнения.

Функция возвращает массив Numpy, содержащий в качестве элементов расчитанное евклидово расстояние,порядок элементов соответствует порядку массива лиц.

Пример:

```
# загружаем 3 фото
image1 = fr.load_image_file("D:/01.jpg")
image2 = fr.load_image_file("D:/02.jpg")
image3 = fr.load_image_file("D:/03.jpg")
# кодируем лица
code1 = fr.face_encodings(image1)[0]
code2 = fr.face_encodings(image2)[0]
code3 = fr.face_encodings(image3)[0]
# создаем список кодов для сравнения
codes=[code1, code2]
# получаем евклидовы расстояния
fr.face_distance(codes,code3)
```

Примеры работы с библиотекой Face_Recognition

```
# Библиотека dlib содержит реализацию алгоритмов глубокого метрического
обучения.
# Установите ее из оболочки или через Anaconda
# conda install -c conda-forge dlib
# Библиотека face_recognition включает в себя функции распознавания лиц dlib
и является надстройкой над ней
!pip install face_recognition

# импорт
import PIL.Image
import PIL.ImageDraw
import face_recognition as fr

import matplotlib.pyplot as plt

# загрузка файла
img = fr.load_image_file('D:/3.jpg')

# просмотр
print(img)
plt.imshow(img)

# поиск лиц на изображении
face_loc = fr.face_locations(img)

# получаем количество лиц
print(len(face_loc))

# рисуем крадраты на лицах
pil_image = PIL.Image.fromarray(img)
# цикл по лицам (по массиву)
for face_location in face_loc:
    # получаем координаты
    top,right,bottom,left =face_location
    # рисуем прямоугольник
    draw_shape = PIL.ImageDraw.Draw(pil_image)
    draw_shape.rectangle([left, top, right, bottom],outline="red")

# выводим изображение
plt.imshow(pil_image)

# загружаем изображения с лицами для сравнения (распознавания лица)
```

```
img_01 = fr.load_image_file("D:/01.jpg")
img_02 = fr.load_image_file("D:/02.jpg")
img_03 = fr.load_image_file("D:/03.jpg")

# кодировка лиц
e_img_01 = fr.face_encodings(img_01)[0]
e_img_02 = fr.face_encodings(img_02)[0]
e_img_03 = fr.face_encodings(img_03)[0]

# сравнение лиц 1 и 2 фото
if fr.compare_faces([e_img_01], e_img_02)[0] == True:
    print ("1 и 2 фото одного человека")
else:
    print ("1 и 2 фото разные люди")

# сравнение лиц 1 и 3 фото
if fr.compare_faces([e_img_01], e_img_03)[0] == True:
    print ("1 и 3 фото одного человека")
else:
    print ("1 и 3 фото разные люди")

# получение координат элементов лица
fl_list = fr.face_landmarks(img_01)
print(fl_list)

# выводим координаты элементов лица
pil_image = PIL.Image.fromarray(img_01)
d = PIL.ImageDraw.Draw(pil_image)
# Просматривать элементы в списке
for fl in fl_list:
    # Распечатать положение каждой черты лица на картинке
    for facial_feature in fl.keys():
        print ("Здесь находятся точки черт лица  - {}: {}"
               .format(facial_feature, fl[facial_feature]))

# Изменить брови
d.polygon(fl['left_eyebrow'], fill=(68, 54, 39, 128))
d.polygon(fl['right_eyebrow'], fill=(68, 54, 39, 128))
d.line(fl['left_eyebrow'], fill=(68, 54, 39, 150), width=5)
d.line(fl['right_eyebrow'], fill=(68, 54, 39, 150), width=5)
pil_image.show()
```

Тема 4. Рекомендательные системы и библиотека Surprise

Задание лабораторной работы

Цель работы: получение практических навыков построения рекомендательных систем на языке Python с использованием библиотеки Surprise.

Задание: используя программу Jupiter Notebook, язык программирования Python, библиотеку Surprise и др.:

1) загрузить набор данных согласно варианту, преобразовать данные в случае необходимости в соответствующий вид,

2) использовать метод согласно варианту для получения рекомендаций (прогнозных рейтингов),
3) получить значения оценок модели прогноза и интерпретировать результат,
4) вывести запрашиваемый в варианте результат (написать функцию с соответствующими входными параметрами и выводом, привести в отчёте 3 результата вызова функции с разными параметрами).

Отчёт по лабораторной работе должен содержать:

6. Фамилию и номер группы учащегося, задание, вариант.
7. Описание полученного набора данных.
8. Полное описание метода из варианта (алгоритм/формулы, выдаваемые значения, их интерпретация).
9. Пример вычислений (в ручную пошагово) по данным и методу из варианта (обязательно).
10. Скрины выполнения программы.
11. Интерпретация результатов (объяснение на конкретных данных)
12. Код с комментариями.

Варианты

Вариант	**Набор данных (закачать любой подходящий набор данных с ресурса)**	**Метод прогноза**	**Вывод**
1	***MovieLens 25M Dataset*** – набор рейтинговых данных с веб-сайта MovieLens, который описывает 5-звездочные рейтинги и действия с произвольным тегированием по более 60 тысячам фильмов от 1,5 миллионов пользователем с 1995 по 2019 годы. https://grouplens.org/datasets/movielens/	random_pred.NormalPredictor	Топ-10 рекомендованных объектов (товаров) по пользователю
2	***Netflix Prize*** - многовариантный датасет временных рядов, который использовался в конкурсе Netflix Prize с рейтингами примерно 100 миллионов фильмов. В наборе данных более 480000 пользователей, каждый из которых промаркирован уникальным целочисленным идентификатором. http://academictorrents.com/details/9b13183dc4d60676b773c9e2cd6de5e5542cee9a	baseline_only.BaselineOnly	5 наиболее похожих пользователей для заданного пользователя
3	***Book-Crossing*** – датасет с рейтингами около 300 тысяч миллионов книг и обезличенными демографическими	knns.KNNBasic	Топ-10 товаров (объектов),

	данными о более 250 тысячах их читателей. http://www2.informatik.uni-freiburg.de/~cziegler/BX/		конкретного пользователя (по реальному рейтингу)
4	***Amazon Review Data*** – многомиллионный набор обзоров, рейтингов и метаданных продуктов (описание, категория, цена, бренд, характеристики, фото), а также данные о просмотре ссылок. https://nijianmo.github.io/amazon/index.html	knns.KNNWithMeans	Топ-10 рекомендованных объектов (товаров) по пользователю
5	***REKKO CHALLENGE*** – набор данных от онлайн-кинотеатра ОККО для конкурса по разработке рекомендательных систем 2019 года. https://boosters.pro/championship/rekko_challenge/data	knns.KNNBaseline	5 наиболее похожих пользователей для заданного пользователя
6	***LastFM*** – датасет содержит информацию о социальных сетях, тегах и прослушивании музыкальных исполнителей от 2 тысяч пользователей онлайн-музыки Last.fm. https://files.grouplens.org/datasets/hetrec2011/	matrix_factorization.SVD	Топ-10 товаров (объектов), конкретного пользователя (по реальному рейтингу)
7	***Social Network Influencer*** – датасет Peerindex, который включает стандартную задачу изучения парных предпочтений. Здесь каждая точка данных описывает двух человек и предварительно рассчитанные стандартизованные функции на основе активности в Twitter: объем взаимодействий, количество подписчиков и пр. для каждого человека. https://www.kaggle.com/c/predict-who-is-more-influential-in-a-social-network/data	matrix_factorization.SVDpp	Топ-10 рекомендованных объектов (товаров) по пользователю
8	***Million Song Dataset*** – набор звуковых фич и метаданных для миллиона современных музыкальных треков от Echo Nest. http://millionsongdataset.com/	matrix_factorization.NMF	5 наиболее похожих пользователей для заданного пользователя
9	***Free Music Archive (FMA)*** – набор легальных аудиозаписей для задач анализа музыки - просмотр, поиск и организация коллекций. https://github.com/mdeff/fma	slope_one.SlopeOne	Топ-10 товаров (объектов), конкретного пользователя (по реальному рейтингу)
10	***Steam Video Games*** - набор данных о действиях	co_clustering.CoClus	Топ-10

	пользователей и их характеристиках от самого популярного хаба видеоигр, PC Gaming Steam https://www.kaggle.com/tamber/steam-video-games/data	tering	рекомендованных объектов (товаров) по пользователю
11	***Ta-Feng*** – набор данных о покупках от ACM RecSys по 23+ тысяч товаров, от продуктов питания и канцелярских товаров до мебели. http://www.bigdatalab.ac.cn/benchmark/bm/dd?data=Ta-Feng	random_pred.NormalPredictor	5 наиболее похожих пользователей для заданного пользователя
12	***Beiren*** – данные о реальных покупках более миллиона человек в супермаркетах Китая за период с 2012 по 2013 год. http://www.bigdatalab.ac.cn/benchmark/bm/dd?data=Beiren	baseline_only.BaselineOnly	Топ-10 товаров (объектов), конкретного пользователя (по реальному рейтингу)
13	***MovieLens 25M Dataset*** – набор рейтинговых данных с веб-сайта MovieLens, который описывает 5-звездочные рейтинги и действия с произвольным тегированием по более 60 тысячам фильмов от 1,5 миллионов пользователем с 1995 по 2019 годы. https://grouplens.org/datasets/movielens/	knns.KNNBasic	Топ-10 рекомендованных объектов (товаров) по пользователю
14	***Jester*** - Анонимные данные о рейтингах шуток (анекдотов) из системы Jester. https://goldberg.berkeley.edu/jester-data/	knns.KNNWithMeans	5 наиболее похожих пользователей для заданного пользователя
15	***REKKO CHALLENGE*** – набор данных от онлайн-кинотеатра ОККО для конкурса по разработке рекомендательных систем 2019 года. https://boosters.pro/championship/rekko_challenge/data	knns.KNNBaseline	Топ-10 товаров (объектов), конкретного пользователя (по реальному рейтингу)
16	***MovieLens 25M Dataset*** – набор рейтинговых данных с веб-сайта MovieLens, который описывает 5-звездочные рейтинги и действия с произвольным тегированием по более 60 тысячам фильмов от 1,5 миллионов пользователем с 1995 по 2019 годы. https://grouplens.org/datasets/movielens/	matrix_factorization.SVD	Топ-10 рекомендованных объектов (товаров) по пользователю
17	***Netflix Prize*** - многовариантный датасет временных рядов, который использовался в конкурсе Netflix Prize с рейтингами примерно 100 миллионов фильмов. В	matrix_factorization.SVDpp	5 наиболее похожих пользователей

	наборе данных более 480000 пользователей, каждый из которых промаркирован уникальным целочисленным идентификатором. http://academictorrents.com/details/9b13183dc4d60676b773c9e2cd6de5e5542cee9a		для заданного пользователя
18	***Book-Crossing*** – датасет с рейтингами около 300 тысяч миллионов книг и обезличенными демографическими данными о более 250 тысячах их читателей. http://www2.informatik.uni-freiburg.de/~cziegler/BX/	matrix_factorization.NMF	Топ-10 товаров (объектов), конкретного пользователя (по реальному рейтингу)
19	***MovieLens 25M Dataset*** – набор рейтинговых данных с веб-сайта MovieLens, который описывает 5-звездочные рейтинги и действия с произвольным тегированием по более 60 тысячам фильмов от 1,5 миллионов пользователем с 1995 по 2019 годы. https://grouplens.org/datasets/movielens/	slope_one.SlopeOne	Топ-10 рекомендованных объектов (товаров) по пользователю
20	***Netflix Prize*** - многовариантный датасет временных рядов, который использовался в конкурсе Netflix Prize с рейтингами примерно 100 миллионов фильмов. В наборе данных более 480000 пользователей, каждый из которых промаркирован уникальным целочисленным идентификатором. http://academictorrents.com/details/9b13183dc4d60676b773c9e2cd6de5e5542cee9a	co_clustering.CoClustering	5 наиболее похожих пользователей для заданного пользователя

* При несогласованности указаний в варианте, сделать соответствующие пояснения и изменения условий.

Методические указания по выполнению лабораторной работы

Библиотека open source Surprise (Simple Python Recommended System Engine) позволяет создавать и тестировать системы рекомендаций на базе различных алгоритмов машинного обучения (прогнозирования рейтинга). Для построения системы необходимы данных об оценках пользователей для конкретных товаров (объектов).

Страница проекта на github: https://github.com/NicolasHug/Surprise

Документация по библиотеке: http://surpriselib.com/

Установка библиотеки

Для установки используйте команду (если используете ее в блокноте поставьте ! в начале строки):

```
pip install scikit-surprise
```

Использование библиотеки

Библиотека Surprise содержит (Рисунок 20):

- классы: Trainset, Reader;
- модули: dataset (набор данных), accuracy (критерии оценки), similarities (метрики сходства), dump (дамп результатов алгоритма);
- пакеты: prediction_algorithms (алгоритмы предсказания), model_selection (модель выбора / кросс-валидации).

Для построения рекомендательной системы необходимо (Рисунок 21):

1) Загрузить данные (для чтения используется класс Reader, для хранения модуль dataset);
2) Выбрать алгоритм предсказания (прогноза) и определить его параметры (пакет prediction_algorithms, алгоритмы, а также модуль similarities и метрики сходства.
3) Разбить при необходимости набор на тестовое и обучающее множества и применить к ним выбранный алгоритм (model_selection),
4) Оценить полученную прогнозную модель (модуль accuracy и критерии оценки Таблица 6),
5) Получить рекомендации по запросу на базе построенной модели (model_selection).

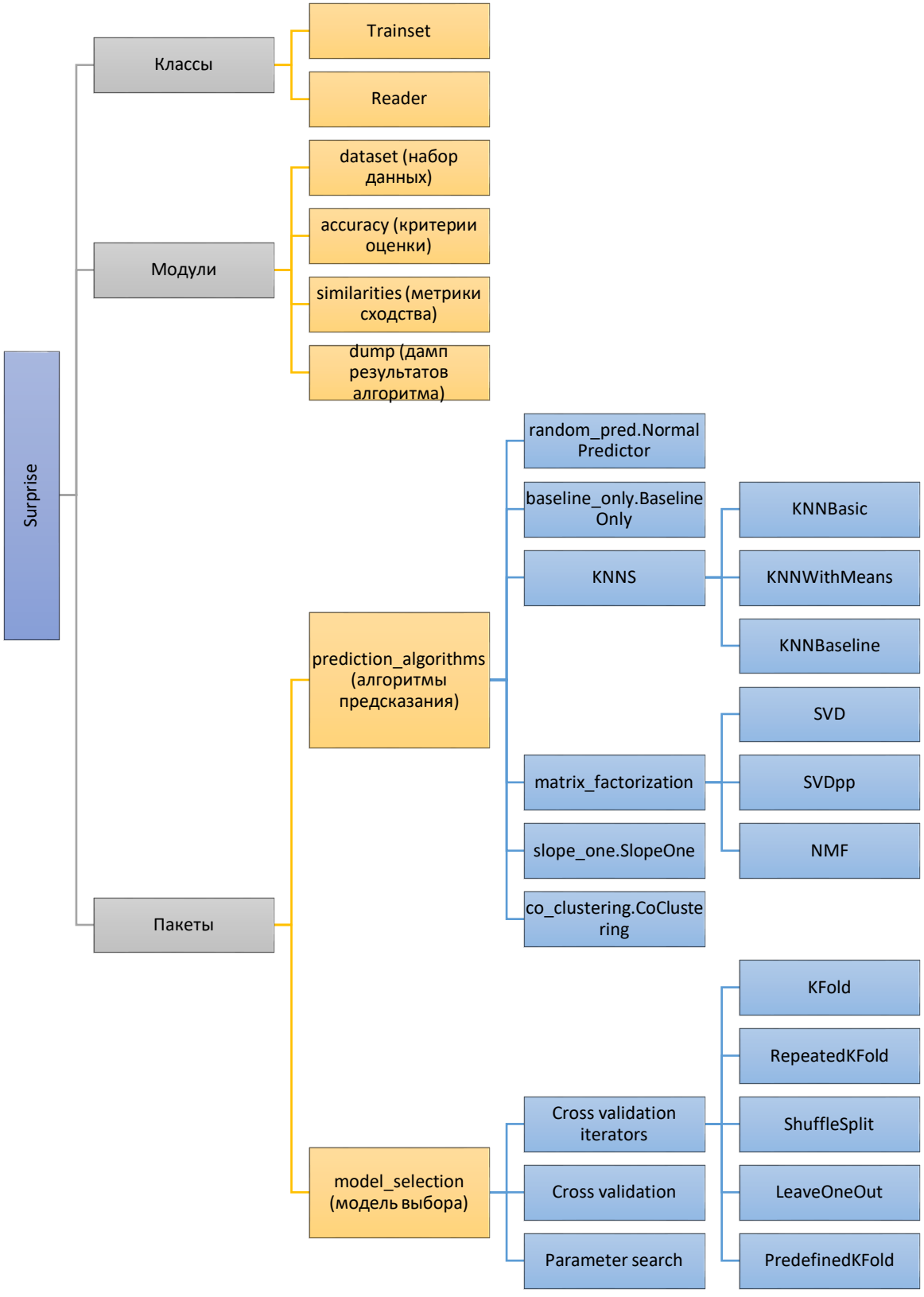


Рисунок 20 – Структура библиотеки

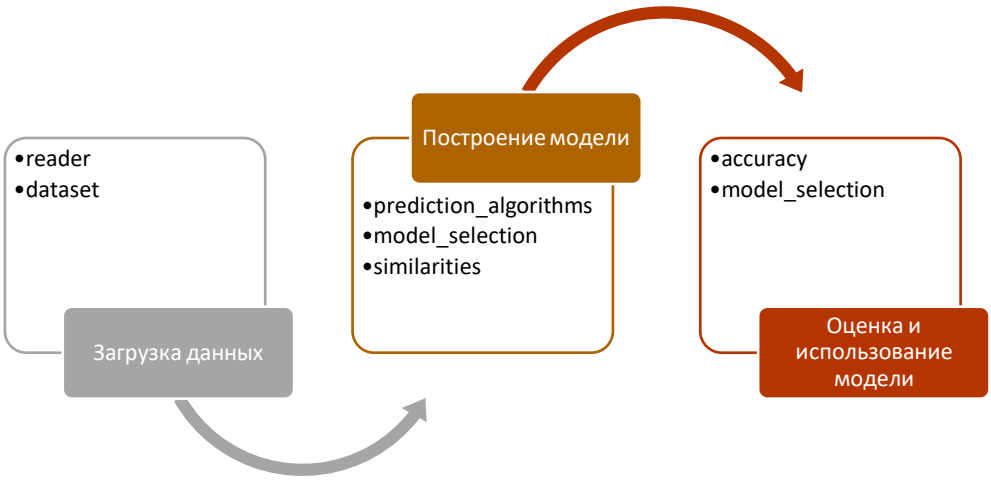


Рисунок 21 – Схема использования элементов библиотеки

Загрузка данных

Данные при использовании библиотека можно:

- воспользоваться встроенными данными,
- загрузить данные из внешнего источника (например файла CSV)
- загрузить данные из DataFrame Pandas (бывает полезно, если необходима подготовка данных).

Для использования данных нужно воспользоваться элементами библиотеки Dataset и Reader.

Пример загрузки встроенного набора данных:

```
from surprise import Dataset
#  Загружать набор данных movielens по умолчанию
data = Dataset.load_builtin('ml-100k')
```

Пример загрузки данных из файла:

```
from surprise import Dataset
from surprise import Reader
import os
#  Укажите путь к файлу
file_path = os.path.expanduser('data.csv')
#  Скажите читателю, какой формат текста
reader = Reader(line_format='user item rating', sep=',')
#  Загрузить данные
data = Dataset.load_from_file(file_path, reader=reader)
```

Пример загрузки данных из DataFrame Pandas:

```
import pandas as pd
from surprise import Dataset
from surprise import Reader
# считываем  данные из файла
fix_df1 = pd.read_csv('d://ratings.csv')
# загружаем данные из набора Pandas в набор Surprise
reader = Reader(rating_scale=(0, 5))
data = Dataset.load_from_df(fix_df[['userId', 'movieId', 'rating']], reader)
```

Построение модели

При построении модели необходимо определить, какой алгоритм будет строить предсказательную модель и как он будет работать с данными. Surprise предоставляет различные инструменты для запуска процедур перекрестной проверки и поиска лучших параметров для алгоритма прогнозирования. Библиотека позволят осуществлять 3 способа, реализованных в пакете model_selection:

- автоматизированная кросс-валидация (класс surprise.model_selection.validation.cross_validate);
- итерационная крос-валиадция (модуль model_selection.split);
- параметрический поиск (класс surprise.model_selection.search.GridSearchCV).

Подробнее см. https://surprise.readthedocs.io/en/stable/model_selection.html

Наиболее простой способ – это использование метода cross_validate с одним из алгоритмов построения модели из пакета prediction_algorithms (Таблица 4).

Таблица 4 – Алгоритмы предсказания (рейтинга)

Имя класса алгоритма	**Объяснение**
random_pred.NormalPredictor	Случайно дать прогнозируемое значение в соответствии с характеристиками распределения обучающего набора
baseline_only.BaselineOnly	Учитывая пользователя и предмет, дать оценку на основе базовой линии
knns.KNNBasic	Самая базовая совместная фильтрация
knns.KNNWithMeans	Внедрение совместной фильтрации с учетом среднего значения рейтинга каждого пользователя
knns.KNNBaseline	Совместная фильтрация с учетом базовых рейтингов
matrix_factorization.SVD	Реализация SVD
matrix_factorization.SVDpp	SVD ++, то есть LFM + SVD
matrix_factorization.NMF	Совместная фильтрация на основе матричной декомпозиции
slope_one.SlopeOne	Простой, но точный алгоритм совместной фильтрации
co_clustering.CoClustering	Алгоритм совместной фильтрации, основанный на совместной кластеризации

Пример:

```
### Использовать NormalPredictor
from surprise import NormalPredictor
algo = NormalPredictor()
perf = cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=3)
```

```
### Использовать BaselineOnly
from surprise import BaselineOnly
algo = BaselineOnly()
perf = cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=3)

### Использовать базовую версию совместной фильтрации
from surprise import KNNBasic, evaluate
algo = KNNBasic()
perf = cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=3)

### Использовать среднюю совместную фильтрацию
from surprise import KNNWithMeans, evaluate
algo = KNNWithMeans()
perf = cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=3)

### Использовать базовую линию совместной фильтрации
from surprise import KNNBaseline, evaluate
algo = KNNBaseline()
perf = cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=3)

### Используйте SVD
from surprise import SVD, evaluate
algo = SVD()
perf = cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=3)

### Используйте SVD ++
from surprise import SVDpp, evaluate
algo = SVDpp()
perf = cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=3)

### Используйте NMF
from surprise import NMF
algo = NMF()
perf = cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=3)
```

Для сравнения строк или столбцов матрицы (расчета схожести объектов или пользователей) могут быть использованы метрики модуля similarities (Таблица 5).

Таблица 5 - Метрика сходства

Метрика	Метрическое описание
cosine	Вычислите косинусное сходство между всеми парами пользователей (или предметов).
msd	Рассчитайте среднеквадратичную разницу между всеми пользователями (или предметами).
pearson	Рассчитайте коэффициент корреляции Пирсона между всеми парами пользователей (или предметов).
pearson_baseline	Рассчитайте (уменьшенный) коэффициент корреляции Пирсона между всеми парами пользователей (или предметов), используя базовую линию для центрирования, а не для среднего значения.

Пример:

```
#  Загружать набор данных movielens по умолчанию
data = Dataset.load_builtin('ml-100k')
trainset = data.build_full_trainset()
#Используйте метод pearson_baseline для вычисления сходства.
```

```
#Ложное вычисление сходства на основе элемента.
#Этот пример - сходство между фильмами
sim_options = {'name': 'pearson_baseline', 'user_based': False}
#Использование алгоритма KNNBaseline
algo = KNNBaseline(sim_options=sim_options)
#Тренировочная модель
algo.fit(trainset)
```

Оценка модели

Для определения насколько модель обучена необходимо ее оценить и проверить на тестовом множестве (если выполнялось разделение набора данных). В библиотеке используются критерии оценки (Таблица 6), реализованные в модуле accuracy.

Таблица 6 - Критерии оценки

Критерии	**Название**
rmse	Рассчитать RMSE (среднеквадратическая ошибка)
mae	Рассчитать MAE (средняя абсолютная ошибка)
fcp	Рассчитать FCP (оценка координационных пар)

Пример

```
rom surprise import SVD
from surprise import Dataset
from surprise.model_selection import KFold
from surprise import accuracy

data = Dataset.load_builtin('ml-100k')
kf = KFold(n_splits=3)
algo = SVD()
for trainset,testset in kf.split(data):
    algo.fit(trainset)
    predictions = algo.test(testset)
    accuracy.rmse(predictions, verbose=True)
```

Пример кода

1) Пример использования разных алгоритмов прогноза

```
# загружаем библиотеки
import pandas as pd
from surprise import Dataset
from surprise import Reader
from surprise.model_selection import cross_validate
from surprise import NormalPredictor
from surprise import SVD
from surprise import accuracy

# считываем  данные из файла
fix_df1 = pd.read_csv('d://ratings.csv')
#ограничиваем набор для учебных целеей
fix_df=fix_df1[:10000]
# загружаем данные из набора Pandas в набор Surprise
reader = Reader(rating_scale=(0, 5))
```

```
data = Dataset.load_from_df(fix_df[['userId', 'movieId', 'rating']], reader)
# применить алгоритм SVD для загруженного набора данных и расчитать оценки
RMSE, MAE
cross_validate(SVD(), data, measures=['RMSE', 'MAE'], cv=5, verbose=True)
# применить алгоритм NormalPredictor к набору данных
cross_validate(NormalPredictor(), data, cv=2)

# делим на обучающую и тестовую выборки
trainset, testset = train_test_split(data, test_size=.25)
# применить алгоритм SVD
al=SVD().fit(trainset)
out2 = al.test(testset)
accuracy.rmse(out2)

# применить алгоритм SVD с указанием параметров
al2=SVD(n_factors=80, n_epochs=20, lr_all=0.005, reg_all=0.2)
al2.fit(trainset)
out4 = al2.test(testset)
df_pred = pd.DataFrame(out4, columns=['user_id', 'isbn', 'actual_rating',
'pred_rating', 'details'])
df_pred['impossible'] = df_pred['details'].apply(lambda x:
x['was_impossible'])
df_pred['pred_rating_round'] = df_pred['pred_rating'].round()
df_pred['abs_err'] = abs(df_pred['pred_rating'] - df_pred['actual_rating'])
df_pred.drop(['details'], axis=1, inplace=True)
```

2) Пример с выводом рекомендованных значений для всех пользователей

```
from collections import defaultdict

from surprise import SVD
from surprise import Dataset
from surprise import Reader

import pandas as pd

def get_top_n(predictions, n=10):
    #top-N рекомендация для каждого пользователя из набора.
    top_n = defaultdict(list)
    for uid, iid, true_r, est, _ in predictions:
        top_n[uid].append((iid, est))
    for uid, user_ratings in top_n.items():
        user_ratings.sort(key=lambda x: x[1], reverse=True)
        top_n[uid] = user_ratings[:n]
    return top_n


# считываем  данные из файла
fix_df = pd.read_csv('d://ratings.csv')

# загружаем данные из набора Pandas в набор Surprise
reader = Reader(rating_scale=(0, 5))
data = Dataset.load_from_df(fix_df[:10000][['userId', 'movieId', 'rating']],
reader)

trainset = data.build_full_trainset()
algo = SVD()
algo.fit(trainset)

# Вычисляем прогнозные значения для тренировочной выборки
testset = trainset.build_anti_testset()
predictions = algo.test(testset)
```

```
top_n = get_top_n(predictions, n=10)

# Выводим рекомендации
for uid, user_ratings in top_n.items():
    print(uid, [iid for (iid, _) in user_ratings])
```

3) Пример использования метод pearson_baseline для вычисления сходства[4]

```
from __future__ import (absolute_import, division, print_function,
                        unicode_literals)
import os
import io
from surprise import KNNBaseline
from surprise import Dataset

import logging

logging.basicConfig(level=logging.INFO,
                    format='%(asctime)s          %(filename)s[line:%(lineno)d]
%(levelname)s %(message)s',
                    datefmt='%a, %d %b %Y %H:%M:%S')


#  Обучение рекомендуемой модели Шаги: 1
def getSimModle():
    #  Загружать набор данных movielens по умолчанию
    data = Dataset.load_builtin('ml-100k')
    trainset = data.build_full_trainset()
    #Используйте метод pearson_baseline для вычисления сходства. Ложное
вычисление сходства на основе элемента. Этот пример - сходство между фильмами
    sim_options = {'name': 'pearson_baseline', 'user_based': False}
    ## Использование алгоритма KNNBaseline
    algo = KNNBaseline(sim_options=sim_options)
    #Тренировочная модель
    algo.fit(trainset)
    return algo


#  Получить взаимное сопоставление идентификатора с именем Шаг: 2
def read_item_names():
    """
        Получить сопоставление названия фильма с идентификатором фильма и
идентификатора фильма с названием фильма
    """
    file_name = (os.path.expanduser('~') +
                 '/.surprise_data/ml-100k/ml-100k/u.item')
    rid_to_name = {}
    name_to_rid = {}
    with io.open(file_name, 'r', encoding='ISO-8859-1') as f:
        for line in f:
            line = line.split('|')
            rid_to_name[line[0]] = line[1]
            name_to_rid[line[1]] = line[0]
    return rid_to_name, name_to_rid


#  Порекомендуйте похожие фильмы по мотивам ранее обученной модели. Шаги: 3
def showSimilarMovies(algo, rid_to_name, name_to_rid):
    #  Получить raw_id фильма История игрушек (1995)
    toy_story_raw_id = name_to_rid['Toy Story (1995)']
    logging.debug('raw_id=' + toy_story_raw_id)
    #Преобразуйте raw_id фильма во внутренний идентификатор модели.
```

[4] https://russianblogs.com/article/8373229539/

```
    toy_story_inner_id = algo.trainset.to_inner_iid(toy_story_raw_id)
    logging.debug('inner_id=' + str(toy_story_inner_id))
    #Получить рекомендуемые фильмы через модель здесь 10
    toy_story_neighbors = algo.get_neighbors(toy_story_inner_id, 10)
    logging.debug('neighbors_ids=' + str(toy_story_neighbors))
    #Преобразовать внутренний идентификатор модели в фактический
идентификатор фильма
    neighbors_raw_ids = [algo.trainset.to_raw_iid(inner_id) for inner_id in
toy_story_neighbors]
    #По списку идентификаторов фильмов или списку рекомендаций по фильмам
    neighbors_movies = [rid_to_name[raw_id] for raw_id in neighbors_raw_ids]
    print('The 10 nearest neighbors of Toy Story are:')
    for movie in neighbors_movies:
        print(movie)


if __name__ == '__main__':
    #  Получить взаимное сопоставление от идентификатора к имени
    rid_to_name, name_to_rid = read_item_names()

    #  Модель рекомендаций по обучению
    algo = getSimModle()

    ## Показать похожие фильмы
    showSimilarMovies(algo, rid_to_name, name_to_rid)
```

Тема 5. Использование библиотеки Natasha для базовых задач NLP

Задание лабораторной работы

Цель работы: получение практических навыков использования генетических алгоритмов на языке Python с использованием библиотеки Natasha.

Задание: используя программу Jupiter Notebook, язык программирования Python, библиотеку Natasha реализовать предварительную обработку текста на русском языке, выполнив следующие задачи:

Отчёт по лабораторной работе должен содержать:

13. Фамилию и номер группы учащегося, задание, вариант.
14. Алгоритм решения задачи.
15. Результаты обработки текста.
16. Код.
17. Обрабатываемый текст на русском языке (найти подходящий или сгенерировать согласно заданию).

Варианты

№	Текст	Задание

1	Любой художественный рассказ	Извлечь все прилагательные из текста и для каждого прилагательного вывести список существительных, с которыми оно употреблялось (в нормализованном виде).
2	Любой художественный рассказ	Подсчитать количество предложений, слов, глаголов, существительных, сколько уникальных глаголов и существительных в тексте, вывести их списки.
3	https://histrf.ru/read/biographies/ivan-iv-groznyi	Извлечь все персоны и сопоставить им все глаголы, с которыми они были связаны (в нормальной форме), подсчитать частоту связанных глаголов.
4	https://ria.ru/20130304/925668903.html	Извлечь «тройки» дата – глагол – персона (упомянутые в одном предложении).
5	http://inmotion.live/notes/mysql-story/	Сопоставить организации и персоны.
6	Любой художественный рассказ	Подсчитать для каждой части речи, сколько уникальных слов было в тексте.
7	В тексте должно быть упоминание не менее 5 валют.	Подсчитать общую сумму денежных средств упомянутых в тексте с учётом курса валют.
8	https://www.rusempire.ru/istoriya-rossii-kratko.html	Для каждого века сделать список персон.
9	https://www.rusempire.ru/istoriya-rossii-kratko.html	Для каждого десятилетия сделать список локаций.
10	https://www.rusempire.ru/istoriya-rossii-kratko.html	Найти все уникальные имена и отчества.

Методические указания по выполнению лабораторной работы

Проект Natasha[5] – набор инструментов для NLP русских текстов, выполняет задачи сегментирования, лемматизации, морфологического и синтаксического анализа, embeddings, извлечения именованных сущностей и фактов.

В репозиторий Natasha (Рисунок 22) входит несколько проектов и они объединяются под единым интерфейсом:

[5] https://github.com/natasha

- Slovnet — deep learning моделирование для обработки естественного русского языка
- Yargy-парсер — извлечение структурированное информации из текстов на русском языке с помощью грамматик и словарей
- Razdel — сегментация русскоязычного текста на токены и предложения
- Navec — компактные эмбеддинги для русского языка
- Corus — коллекция ссылок на публичные русскоязычные датасеты + функции для загрузки
- Ipymarkup — визуализация разметки именованных сущностей и синтаксических связей
- Naeval — количественное сравнение систем для русскоязычного NLP
- Nerus — большой синтетический датасет с разметкой морфологии, синтаксиса и именованных сущностей

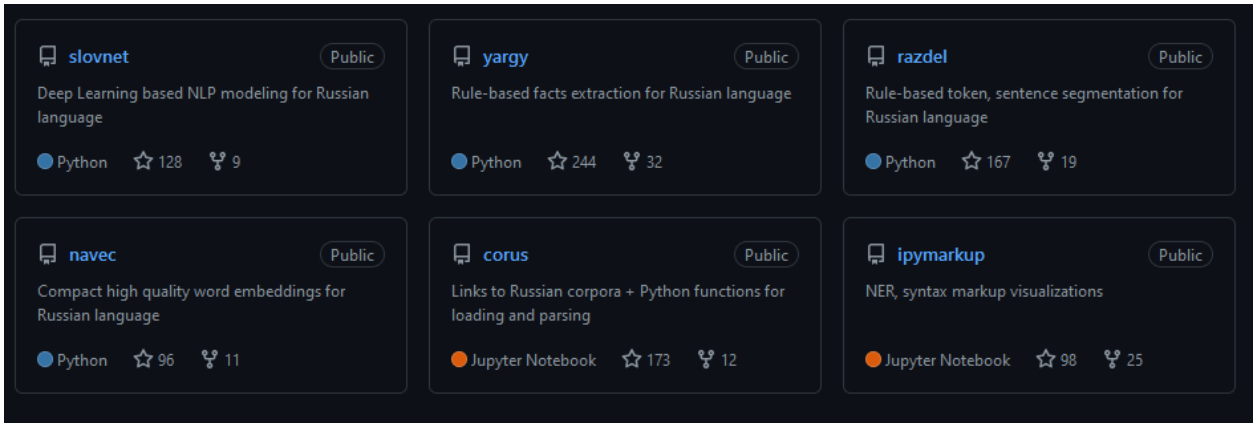


Рисунок 22 – Описание репозитория проекта на GitHub

Полные список можно посмотреть по ссылке https://github.com/orgs/natasha/repositories.

Информацию о каждом из проектов можно получить https://habr.com/ru/post/516098/.

Подключение библиотеки Natasha

Установить библиотеку для работы можно с помощью команды:

```
pip install natasha
```

Библиотека содержит несколько модулей (методов), решающие различные задачи (Таблица 7).

Таблица 7 – Модули библиотеки

Модуль	Назначение	Описание
Doc	Модуль представления текстового документа	Хранит текст и все атрибуты элементов текста
Segmenter	Модуль токинизации (сегментирования)	разбивает текст на токены: слова и предложения
MorphVocab	Модуль морфологического разбора	Реализует морфологический разбор и визуализирует

		атрибутику токинов
NewsMorphTagger	Модуль морфологического разбора (атрибутирование токенов)	Атрибутирует токены морфологическими признаками (часть речи, число и т.д.)
NewsEmbedding	Модуль embedding	Переводит текст в векторное представление для дальнейшей обработки моделями (методами) NLP
NewsSyntaxParser	Модуль синтаксического разбора	Реализует разбор связей слов в предложении и визуализирует их
NewsNERTagger	Модуль извлечения именованных сущностей	Атрибутирует токены признаками именованных сущностей (PER, LOC, ORG)
PER	Модуль разбора персоналий	Реализует разбор персоналий на составляющие (фамилия, имя, отчество)
NamesExtractor	Модуль извлечения именованных сущностей (разметка текста)	Реализует NER, позволяет выделить (визуализировать) сущности в тексте
MoneyExtractor	Модуль извлечения денежных сумм	Извлекает данные о деньгах с учетом различной валюты
AddrExtractor	Модуль извлечения адресов	Извлекает адреса и разбивает их на составляющие
DatesExtractor	Модуль извлечения дат	Извлекает даты записанные в различных форматах

Для подключения библиотеки используем import и указываем те модули, которые необходимы для решения задачи.

Пример:

```
# подключаем библиотеку
from natasha import (
    Segmenter,
    MorphVocab,
    NewsEmbedding,
    NewsMorphTagger,
```

```
    NewsSyntaxParser,
    NewsNERTagger,
    PER,
    NamesExtractor,
    MoneyExtractor,
    AddrExtractor,
    DatesExtractor,
    Doc
)
```

Задача токенизации (сегментации)

Для выполнения анализа необходимо сначала выделить предложения в тексте и слова в предложении. Предложения будут находится в массиве sents, слова – tokens.

Пример:

```
# инициализация компонента для сегментации
segmenter = Segmenter()
# сегментирование текста
doc.segment(segmenter)
# просмотр информации о первом сегменте (предложении)
display(doc.sents[0])
# просмотр информации о первом токине (слове)
display(doc.tokens[0])
```

Задача синтаксического анализа

Для синтаксического разбора необходимо воспользоваться NewsSyntaxParser, на вход которого текст подаётся после процесса embedding. Результат анализа можно визуализировать в виде схемы (Рисунок 23), дуги в которой маркируются зависимостями (Таблица 8, Рисунок 24).

Пример:

```
# инициализация компонента для embedding
emb = NewsEmbedding()
# инициализация компонента для синтаксического разбора
syntax_parser = NewsSyntaxParser(emb)
# выполняем синтаксический раздор для объекта doc (для нашего текста)
doc.parse_syntax(syntax_parser)
# выводим схему синтаксических связей для выбранного предложения
sent.syntax.print()
```

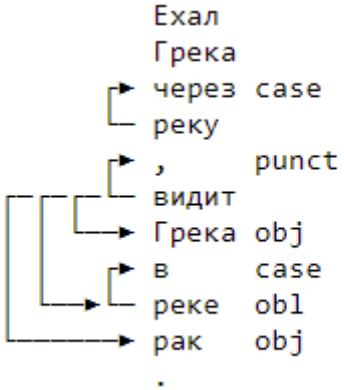


Рисунок 23 – Пример синтаксического разбора

Таблица 8 - Определения типизированных зависимостей Стэнфорда [6]

Обозначение	Название зависимости
CCOMP	clausal complement
APPOS	appositional modifier
ADVMOD	adverb modifier
AGENT	agen
AMOD	adjectival modifier
ADVCL	adverbial clause modifier
DET	determiner
ACOMP	adjectival complement
AUX	auxiliary
AUXPASS	passive auxiliary
CC	coordination
CONJ	conjunct
COP	copula
CSUBJ	clausal subject
CSUBJPASS	clausal passive subject
DEP	dependent
DISCOURSE	discourse element
DOBJ	direct objec
EXPL	expletive
GOESWITH	goes with
IOBJ	indirect object
MARK	marker
MWE	multi-word expression
NEG	negation modifier
NN	noun compound modifier
NPADVMOD	noun phrase as adverbial modifier
NSUBJ	nominal subject
NSUBJPASS	passive nominal subject
NUM	numeric modifier
NUMBER	element of compound number
PARATAXIS	parataxis

[6] https://downloads.cs.stanford.edu/nlp/software/dependencies_manual.pdf

PCOMP	prepositional complement
POBJ	object of a preposition
POSS	possession modifier
POSSESSIVE	possessive modifier
PRECONJ	preconjunct
PREDET	predeterminer
PREP	prepositional modifier
PREPC	prepositional clausal modifier
PRT	phrasal verb particle
PUNCT	punctuation
QUANTMOD	quantifier phrase modifier
RCMOD	relative clause modifier
REF	referent
ROOT	root
TMOD	temporal modifier
VMOD	reduced non-finite verbal modifier
XCOMP	open clausal complement
XSUBJ	controlling subjec

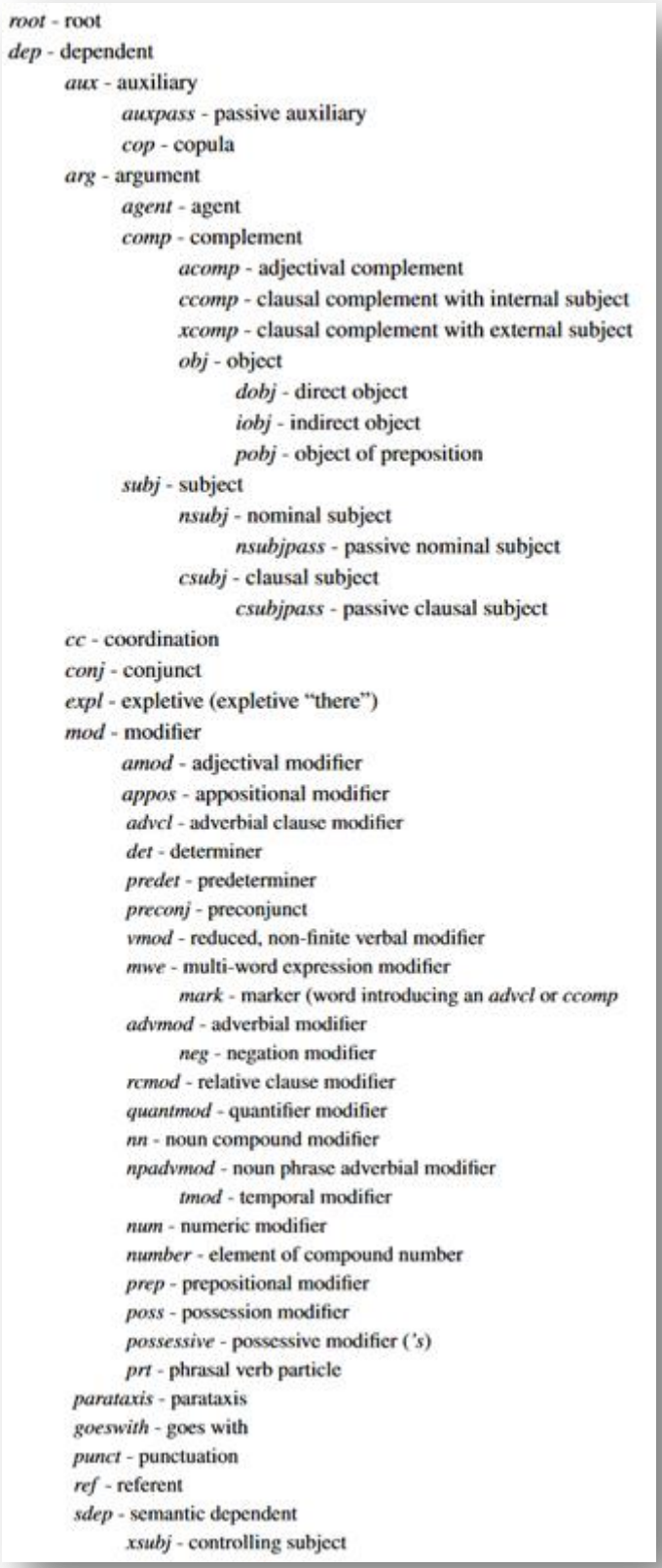


Рисунок 24- Дерево зависимостей

Задача морфологического анализа

Морфологический анализ выполнять надо после синтаксического разбора, он позволяет определить морфологические атрибуты токенов (Таблица 9). Для нахождения этих атрибутов необходимо воспользоваться `NewsMorphTagger`, на вход которого текст подаётся после процесса embedding.

Пример:

```
# инициализация компонента для embedding
emb = NewsEmbedding()
# инициализация компонента для морфологического разбора
morph_tagger = NewsMorphTagger(emb)
doc.tag_morph(morph_tagger)
# выводим результат морфологического разбора для выбранного предложения
sent.morph.print()
```

Пример вывода результатов:

```
  Ехал VERB|Aspect=Imp|Gender=Masc|Mood=Ind|Number=Sing|Tense=Past|VerbForm=Fin|Voice=Act
 Грека PROPN|Animacy=Anim|Case=Gen|Gender=Masc|Number=Sing
 через ADP
  реку NOUN|Animacy=Inan|Case=Acc|Gender=Fem|Number=Sing
     , PUNCT
 видит VERB|Aspect=Imp|Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin|Voice=Act
 Грека PROPN|Animacy=Anim|Case=Acc|Gender=Masc|Number=Sing
     в ADP
  реке NOUN|Animacy=Inan|Case=Loc|Gender=Fem|Number=Sing
   рак NOUN|Animacy=Inan|Case=Acc|Gender=Masc|Number=Sing
     . PUNCT
```

Таблица 9 – Морфологические атрибуты и их значения

Параметр	**Обозначение**	**Обозначение значения**	**Значение**
Часть речи		NOUN	Существительное
		VERB	Глагол
		PROPN	Имя собственное
		ADJ	Прилагательное
		ADV	Наречие
		PRON	Местоимение
		NUM	Числительное
		INTJ	междометие
		ADP	предлог
		CONJ	Союз
		PART	частица
		PUNCT	Знак пунктуации
		SYM	Символ
		X	иное
Род	Gender	MASC	Мужской
		FEM	Женский
		NEUT	Средний
Число	Number	SING	Единственное

		PLUR	Множественное
Время	Tense	PAST	Прошлое
		FUT	Будущее
		PRES	Настоящее
Вид глагола в английском языке	Aspect	IMP	Общий
		PERF	Длительный
Залог	Voice	ACT	Активный
		PASS	Пассивный
Наклонение	Mood	IMP	Повелительное
		INT	Вопросительное
		IND	Изъявительное
Падеж	Case	GEN	Родительный
		INS	Творительный
		ACC	Винительный
		DAT	Дательный
		LOC	Предложный
		NOM	Именительный
Форма глагола	VerbForm	FIN	Личная
		INF	Неличная - инфинитив
		PART	Неличная - причастие
		CONV	Неличная - герундий
Одушевлённость	Animacy	ANIM	Одушевлённый
		INAN	Неодушевлённый
Аббревиатура	Abbr	YES	Аббревиатура

Задача лемматизации

Natasha решает задачу лемматизации с помощью библиотеки Pymorphy2, результатом являются нормализованные токены:

- имена существительные – в именительном падеже единственного числа;
- имена прилагательные – в именительном падеже единственного числа и мужского рода;
- глаголы, а так же причастия и деепричастия – глаголы в исходной форме (в инфинитиве).

Пример:

```
# инициализация компонента для задачи леммизации
morph_vocab = MorphVocab()
# обработка токинов (слов) в цикле
for token in doc.tokens:
    token.lemmatize(morph_vocab)
# вывод результатов (двух полей: изначальный вариант и лемма)
{_.text: _.lemma for _ in doc.tokens}
```

Задача обработки именованных сущностей

Задача извлечения именованных сущностей

Модуль извлечения именованных сущностей NewsNERTagger не зависит от результатов морфологического и синтаксического разбора, его можно использовать отдельно.

Пример:

```
# загружаем модуль
from natasha import NewsNERTagger
# инициируем компонент
ner_tagger = NewsNERTagger(emb)
doc.tag_ner(ner_tagger)
# выводим на экран
doc.ner.print()
```

Подробный пример разбора задачи можно посмотреть https://github.com/mannefedov/compling_nlp_hse_course/blob/master/notebooks/NER.ipynb.

Для задач извлечения именованных сущностей можно создавать свои правила, они пушится непосредственно для парсера Yargy. Пример см. https://nbviewer.jupyter.org/github/natasha/yargy/blob/master/docs/index.ipynb.

Задача нормализации именованных сущностей

Именованные сущности могут быть составными (состоять из нескольких согласованных слов) и их нормализация должна выполнять с учетом всего набора слов (без из разделения).

Пример:

```
# инициализация компонента для embedding
emb = NewsEmbedding()
# инициализация компонента для сегментации
segmenter = Segmenter()
# инициализация компонента для задачи леммизации
morph_vocab = MorphVocab()
# инициализация компонента для извлечения именованных сужностей
ner_tagger = NewsNERTagger(emb)
names_extractor = NamesExtractor(morph_vocab)
# сегментирование текста
doc.segment(segmenter)
# атрибутирование именованных сущностей
doc.tag_ner(ner_tagger)
# нормализация именованных сущностей
for span in doc.spans:
    span.normalize(morph_vocab)
# вывод результатов (двух полей: изначальный вариант и нормализованный)
{_.text: _.normal for _ in doc.spans}
```

Задача разбора персоналий

Для этой задачи нужны модули PER и NamesExtractor. Имена собственные могут буть составными, для русского языка – фамилия, имя, отчество. Можно выполнить разбот таких имен.

Пример:

```
# инициализация компонента для извлечения именованных сущностей
names_extractor = NamesExtractor(morph_vocab)
for span in doc.spans:
    # если сущность является персоналией
    if span.type == PER:
        # разбор имени
        span.extract_fact(names_extractor)
# вывод результатов (двух полей: нормализованного варианта имени и массив с
разбором)
{_.normal: _.fact.as_dict for _ in doc.spans if _.type == PER}
```

Задача извлечения фактов

Кроме локация, именованных сущностей и организация в проекте есть правила для извлечения дат, денежных сумм и адресов.

Извлечение дат

Пример:

```
# инициализация компонента для задачи леммизации
morph_vocab = MorphVocab()
# инициализация компонента для извлечения дат
dates_extractor = DatesExtractor(morph_vocab)
text = '24.01.2017, 2015 год, 2014 г, 1 апреля, май 2017 г., 9 мая 2017 года'
list(dates_extractor(text))
```

Извлечение денежных сумм

Пример:

```
# инициализация компонента для извлечения денежных сумм
money_extractor = MoneyExtractor(morph_vocab)
text = '1 599 059, 38 Евро, 420 долларов, 20 млн руб, 20 т. р., 881 913
(Восемьсот восемьдесят одна тысяча девятьсот тринадцать) руб. 98 коп.'
list(money_extractor(text))
```

Извлечение адресов

Пример:

```
# инициализация компонента для измлечения адрессов
addr_extractor = AddrExtractor(morph_vocab)
lines = [
    'Россия, Вологодская обл. г. Череповец, пр.Победы 93 б',
    '692909, РФ, Приморский край, г. Находка, ул. Добролюбова, 18',
    'ул. Народного Ополчения д. 9к.3'
]
for line in lines:
    display(addr_extractor.find(line))
```

Пример предварительного анализа текста

1) Пример возможностей библиотеки Natasha https://nbviewer.jupyter.org/github/natasha/natasha/blob/master/docs.ipynb

2) Примеры по Yargy «Задача - научиться извлекать из документа ссылка на АПК РФ и т.п.» https://github.com/alexmk7/python_school/blob/master/yargy.ipynb и

https://nbviewer.jupyter.org/github/natasha/yargy/blob/master/docs/cookbook.ipynb

3) Пример кода с разобранными выше задачами.

```
# pip install natasha
# подключаем библиотеку
from natasha import (
    Segmenter,
    MorphVocab,
    NewsEmbedding,
    NewsMorphTagger,
    NewsSyntaxParser,
    NewsNERTagger,
    PER,
    NamesExtractor,
    MoneyExtractor,
    AddrExtractor,
    DatesExtractor,
    Doc
)

# обрабатываемый текст
texts ="""Ехал Грека через реку, видит Грека в реке рак.
    Сидоров Иван Иванович шел по улице Приозерской.
    В 1983 годы родился кот Василек."""
# создаем объект обработки
doc = Doc(texts)

# инициализация компонента для сегментации
segmenter = Segmenter()
# сегментирование текста
doc.segment(segmenter)
# просмотр информации о первом сегменте (предложении)
display(doc.sents[0])
# просмотр информации о первом токине (слове)
display(doc.tokens[0])

# инициализация компонента для embedding
emb = NewsEmbedding()
# инициализация компонента для синтаксического разбора
syntax_parser = NewsSyntaxParser(emb)
# выполняем синтаксический раздор для объекта doc (для нашего текста)
doc.parse_syntax(syntax_parser)
# выбираем первое предложение
sent = doc.sents[0]
# выводим схему синтаксических связей для выбранного предложения
print("\n Синтаксический разбор первого предлложения")
sent.syntax.print()


# инициализация компонента для морфологического разбора
morph_tagger = NewsMorphTagger(emb)
doc.tag_morph(morph_tagger)
# выводим результат морфологического разбора для выбранного предложения
print("\n Морфологический разбор")
sent.morph.print()

# инициализация компонента для задачи леммизации
morph_vocab = MorphVocab()
# обработка токинов (слов) в цикле
for token in doc.tokens:
    token.lemmatize(morph_vocab)
```

```
# вывод результатов (двух полей: изначальный вариант и лемма)
print("\n Лемматизация")
print({_.text: _.lemma for _ in doc.tokens})


# инициализация компонента для извлечения именованных сужностей
ner_tagger = NewsNERTagger(emb)
doc.tag_ner(ner_tagger)
# выводим на экран
print("\n Извлеченные именованные сущности")
doc.ner.print()

# инициализация компонента для извлечения именованных сужностей
names_extractor = NamesExtractor(morph_vocab)
# атрибутирование именованных сущностей
doc.tag_ner(ner_tagger)
# нормализация именованных сущностей
for span in doc.spans:
    span.normalize(morph_vocab)
# вывод результатов (двух полей: изначальный вариант и нормализованный)
print("\n Нормализованные именованные сущности")
print({_.text: _.normal for _ in doc.spans})


# инициализация компонента для извлечения именованных сужностей
names_extractor = NamesExtractor(morph_vocab)
for span in doc.spans:
    # если сущность является персоналией
    if span.type == PER:
        # разбор имени
        span.extract_fact(names_extractor)
# вывод результатов (двух полей: нормализованного варианта имени и массив с
разбором)
print("\n Нормализованные именованные сущности с разбиением на составляющие
имен")
print({_.normal: _.fact.as_dict for _ in doc.spans if _.type == PER})


# инициализация компонента для извлечения дат
dates_extractor = DatesExtractor(morph_vocab)
text = '24.01.2017, 2015 год, 2014 г, 1 апреля, май 2017 г., 9 мая 2017 года'
print("\n Извлеченные даты")
print(list(dates_extractor(text)))

# инициализация компонента для извлечения денежных сумм
money_extractor = MoneyExtractor(morph_vocab)
text = '1 599 059, 38 Евро, 420 долларов, 20 млн руб, 20 т. р., 881 913
(Восемьсот восемьдесят одна тысяча девятьсот тринадцать) руб. 98 коп.'
print("\n Извлеченные деньги")
print(list(money_extractor(text)))


# инициализация компонента для измлечения адрессов
addr_extractor = AddrExtractor(morph_vocab)
lines = [
    'Россия, Вологодская обл. г. Череповец, пр.Победы 93 б',
    '692909, РФ, Приморский край, г. Находка, ул. Добролюбова, 18',
    'ул. Народного Ополчения д. 9к.3'
]
print("\n Извлеченные адреса")
for line in lines:
    display(addr_extractor.find(line))
```

Тема 6. Использование библиотеки PyTorch для реализации искусственных нейронных сетей.

Задание лабораторной работы

Цель работы: получение практических навыков программирования нейронных сетей на языке Python с использованием библиотеки PyTorch.

Задание: используя программу Jupiter Notebook, язык программирования Python, библиотеку PyTorch построить нейронную сеть по варианту и использовать для получения результата.

Работа заключается в:

- Загрузке / генерации данных для обучения НС;
- Построения НС;
- Обучения НС;
- Проверки Нс на тестовых данных;
- Визуализация результата.

Отчёт по лабораторной работе должен содержать:

1. Фамилию и номер группы учащегося, задание, вариант
2. Схему НС (ее слоёв)
3. Описание входных данные
4. Описание алгоритма обучения с учетом варианта (функции потерь, оптимизатора и т.д.)
5. Графики динамики обучения НС.
6. Результат тестирования НС.
7. Код.

Методические указания по выполнению лабораторной работы

Пакет torth.nn

Пакет torch.nn используется для создания нейронных сетей. Он содержит контейнеры для НС, в котором определяются слои, функции потерь, активации, различные методы оптимизации для реализации обучения и т.д.

Пакет nn определяет набор модулей, которые примерно эквивалентны слоям нейронной сети. Модуль принимает входные Tensors и вычисляет выходные Tensors, но может также содержать внутреннее состояние, такое как Tensors, содержащее обучаемые параметры.

Алгоритм работы с НС

1 Подготовка данных для обучения /анализа (обучающая выборка), их преобразование.

2 Выбирается тип НС в зависимости от поставленной задачи (прямого распространения, рекуррентная, сверочная и т.д.), выбирается архитектура сети (количество слоёв, нейронов в слоях, типы слоёв, функции активации), строится модель.

3 Определение функции потерь.

4 Определение оптимизатора.

5 Цикл обучения НС (на примере сети прямого распространения)

- ввод данных и вычисление результата (прямой проход)
- вычисление потери (насколько далёк результат от правильности).
- градиентный спуск и коррекция весов (обратный проход).

6 Запуск / тестирование НС (на тестовой выборке).

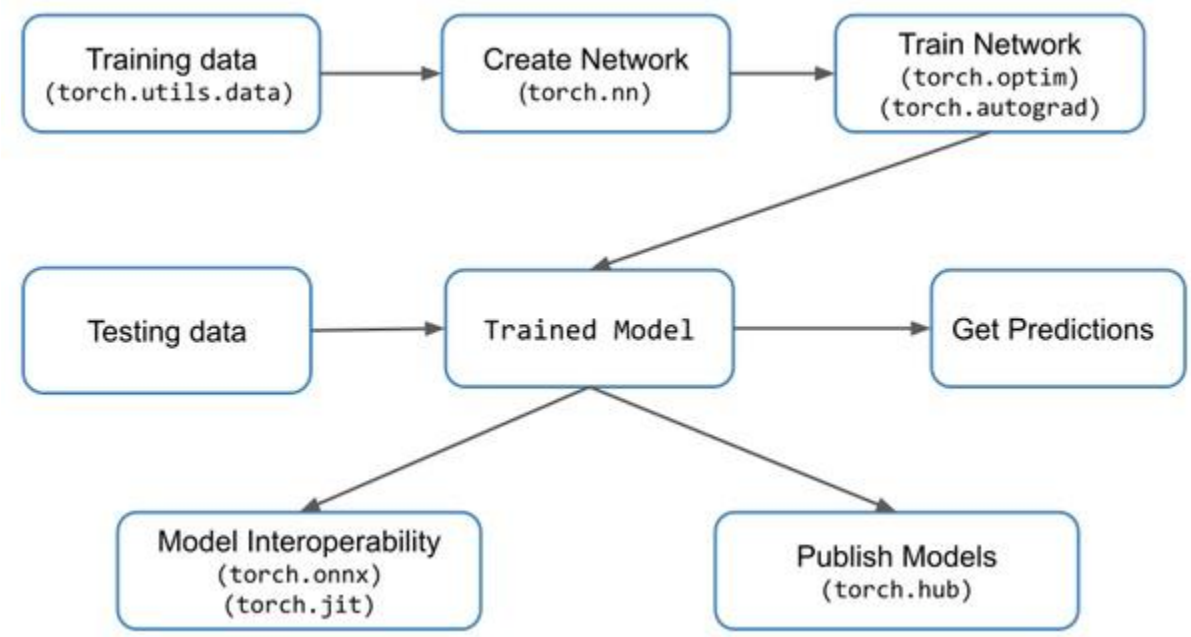


Рисунок 25 – Схема применения модулей PyTorch для обучения НС[7]

Загрузка подготовленного набора данных

PyTorch включает в себя пакет torchvision, который используется для загрузки и подготовки набора данных (https://pytorch.org/docs/master/torchvision/index.html). Он включает в себя две основные функции, а именно Dataset и DataLoader, которые помогают в преобразовании и загрузке набора данных.

[7] https://ai-news.ru/2019/07/pytorch_dlya_nachinaushih_osnovy.html

Dataset построен поверх тензорного типа данных и используется в основном для пользовательских наборов данных. Набор данных используется для чтения и преобразования точки данных из данного набора данных.

Dataset - абстрактный класс, представляющий набор данных. Пользовательский набор данных должен наследовать Dataset и переопределять следующие методы:

- __len__, чтобы len(dataset) возвращал размер набора данных.
- __getitem__ для поддержки индексации, так что dataset[i] может использоваться для получения i-го экземпляра

Основной синтаксис для реализации упомянут ниже:

```
trainset = torchvision.datasets.CIFAR10(root = './data', train = True,
 download = True, transform = transform)
```

Пример:

1)Загрузка набора данных MNIST

```
import torchvision
train_dataset = torchvision.datasets.MNIST(root='g:\\DataForNN2', train=True,
transform=False, download=True)
```

3) Загрузка набора CIFAR10

```
trainset = torchvision.datasets.CIFAR10(root = DATA_PATH, train = True,
download = True, transform = False)
```

4) Загрузка набора STL10

```
torchvision.datasets.STL10(DATA_PATH, split='train', folds=None,
transform=None, target_transform=None, download=True)
```

Далее необходимо создать объекты train_dataset и test_dataset, которые будут последовательно проходить через загрузчик данных. Чтобы создать такие датасеты из данных MNIST, требуется задать несколько аргументов. Первый — путь до папки, где хранится файл с данными для тренировки и тестирования. Логический аргумент train показывает, какой файл из train.pt или test.pt стоит брать в качестве тренировочного сета. Следующий аргумент — transform, в котором мы указываем ранее созданный объект trans, который осуществляет преобразования. Наконец, аргумент загрузки просит функцию датасета MNIST загрузить при необходимости данные из онлайн источника.[8]

[8] https://neurohive.io/ru/tutorial/cnn-na-pytorch/

Таблица 10 – Примеры наборов данных для обучения (полный список см. https://pytorch.org/docs/stable/torchvision/datasets.html)

MNIST	Рукописные цифры 1–9. Подмножество набора данных NIST рукописных символов. Содержит обучающий набор из 60000 тестовых изображений и тестовый набор из 10000.
Fashion-MNIST	Набор данных для MNIST. Содержит изображения предметов моды; например, футболка, брюки, пуловер.
EMNIST	На основе рукописных символов NIST, включая буквы и цифры и разделение для задач классификации классов 47, 26 и 10.
COCO	Более 100 000 изображений, классифицированных в повседневные предметы; например, человек, рюкзак и велосипед. Каждое изображение может иметь более одного класса.
LSUN	Используется для крупномасштабной классификации сцен изображений; например, спальня, мост, церковь.
Imagenet-12	Крупномасштабный набор данных визуального распознавания, содержащий 1,2 миллиона изображений и 1000 категорий. Реализовано с классом ImageFolder, где каждый класс находится в папке.
CIFAR	60 000 цветных изображений с низким разрешением (32 32) в 10 взаимоисключающих классах; например, самолет, грузовик и автомобиль.
STL10	Аналогично CIFAR, но с более высоким разрешением и большим количеством немаркированных изображений.
SVHN	600 000 изображений номеров улиц, полученных из Google Street View. Используется для распознавания цифр в реальных условиях.
PhotoTour	Изучение локальных дескрипторов изображений. Состоит из полутоновых изображений, состоящих из 126 фрагментов, сопровождаемых текстовым файлом дескриптора. Используется для распознавания образов.

Преобразование данных

Функция transform.Compose() находится в пакете torchvision и позволяет выполнять трансформацию набора данных, причём трансформаций может быть несколько и они представляются списком.

Пример:

```
1) Загрузка MNIST с преобразованием
import torchvision
```

```
import torchvision.transforms as transforms
# путь куда грузим
DATA_PATH = 'g:\\DataForNN'
# выполняемое преобразование над набором данных
trans = transforms.Compose([transforms.ToTensor(),
transforms.Normalize((0.1307,), (0.3081,))])
# грузим набор данных тренировочный
train_dataset = torchvision.datasets.MNIST(root=DATA_PATH, train=True,
transform=trans, download=True)
# грузим набор данных тестовый
test_dataset = torchvision.datasets.MNIST(root=DATA_PATH, train=False,
transform=trans)
```

В примере устанавливается преобразование, которое конвертирует входной датасет в PyTorch тензор. PyTorch тензор — особый тип данных, используемый в библиотеке для всех различных операций с данными и весами внутри нейросети. Следующий аргумент в списке Compose() — нормализация. Нейронная сеть обучается лучше, когда входные данные нормализованы так, что их значения находятся в диапазоне от -1 до 1 или от 0 до 1. Чтобы это сделать с помощью нормализации PyTorch, необходимо указать среднее и стандартное отклонение MNIST датасета, которые в этом случае равны 0.1307 и 0.3081 соответственно. У MNIST есть только один канал, но уже для датасета CIFAR с 3 каналами (по одному на каждый цвет из RGB спектра) надо указывать среднее и стандартное отклонение для каждого.

Загрузка данных для тренировки нейронной сети

DataLoader используется, когда есть большой набор данных, и необходимо загрузить данные из Dataset в фоновом режиме, чтобы он был готов и ждал цикла обучения.

DataLoader используется для перемешивания и пакетной обработки данных. Он может использоваться для загрузки данных параллельно с многопроцессорными рабочими. Объект загрузчик данных в PyTorch обеспечивает несколько полезных функций при использовании тренировочных данных:

- Возможность легко перемешивать данные.
- Возможность группировать данные в партии.
- Более эффективное использование данных с помощью параллельной загрузки, используя многопроцессорную обработку.

Синтаксис:

```
trainloader = torch.utils.data.DataLoader(trainset, batch_size = 4,
 shuffle = True)
```

Первый — данные, которые вы хотите загрузить; второй — желаемый размер партии; третий — перемешивать ли случайным образом датасет.

Построение нейронной сети

На базе *nn.Module.* Необходимо наследовать класс *nn.Module* и реализовать методы инициализации *init* и прямого прохода/вычисления *forward.* Синтаксис следующий:

```
# подключаем модуль torch.nn
import torch.nn as nn
# импортируем функции активации
import torch.nn.functional as F
# Model это имя
class Model(nn.Module):
 def __init__(self):
 super(Model, self).__init__()
 …
 def forward(self, x):
 …
```

Внутри функции *__init__* объявляют слои будущей нейронной сети, они могут быть разными по тиру, в зависимости от того, какую нейронную сеть строим, количество и размерность слоёв определяется здесь же. Размерность следующих друг за другом слоев должна быть согласована, сколько выходов в предшествующем, столько входов в последующем.

Пример:

1)Два линейных слоя (нейронная сеть прямого распространения, в которой слои полносвязные), которые имеют вид **W'x+b**, где W — матрица весов размером *(input, output)* и b — вектор смещения размером *output*. Первый слой размерностью (784, 100), второй (100, 10), т.е. выходной вектор НС размерностью 10:

```
class Model(nn.Module):
def __init__(self):
 super(Net, self).__init__()
 self.fc1 = nn.Linear(784, 100)
 self.fc2 = nn.Linear(100, 10)
```

2)Два слоя двумерной свёртки, первый слой имеет 1 входной канал, 20 выходных и размер ядра 5, второй, 20 входных :

```
class Model(nn.Module):
```

```
def __init__(self):
 super(Model, self).__init__()
 self.conv1 = nn.Conv2d(1, 20, 5)
 self.conv2 = nn.Conv2d(20, 20, 5)
```

3) Свёрточная сеть с 2 сверточными слоями и 3 линейными (полносвязными):

```
class Model(nn.Module):
def __init__(self):
 super(Net, self).__init__()
 self.conv1 = nn.Conv2d(1, 6, 3)
 self.conv2 = nn.Conv2d(6, 16, 3)
 self.fc1 = nn.Linear(16 * 6 * 6, 120)
 self.fc2 = nn.Linear(120, 84)
 self.fc3 = nn.Linear(84, 10)
```

4) Свёрточная сеть Conv2d -> MaxPool2d -> Conv2d -> MaxPool2d -> Linear -> Linear.

```
class MNISTConvNet(nn.Module):
 def __init__(self):
 super(MNISTConvNet, self).__init__()
 self.conv1 = nn.Conv2d(1, 10, 5)
 self.pool1 = nn.MaxPool2d(2, 2)
 self.conv2 = nn.Conv2d(10, 20, 5)
 self.pool2 = nn.MaxPool2d(2, 2)
 self.fc1 = nn.Linear(320, 50)
 self.fc2 = nn.Linear(50, 10)
```

5) Рекуррентная нейронная сеть.

```
class RNNModel(nn.Module):
 def __init__(self, input_dim, hidden_dim, layer_dim, output_dim):
 super(RNNModel, self).__init__()
 self.hidden_dim = hidden_dim
 self.layer_dim = layer_dim
 self.rnn = nn.RNN(input_dim, hidden_dim, layer_dim, batch_first=True,
 nonlinearity='relu')
 self.fc = nn.Linear(hidden_dim, output_dim)
```

Метод *forward* используется непосредственно для преобразования входных данных с помощью заданной нейронной сети в ее выходы.

Вычисляемая функция может быть любой сложности, но должна учитывать заданные слои в функции init.

Пример:

1) Определение для линейных слоёв из примеры выше. Функция view() переиндексирует тензор с данными заданным образом, "-1" в качестве первого аргумента функции означает, что количество элементов в первой размерности будет вычислено автоматически. Если исходный тензор x имеет размерность (N, 28, 28), то после x = x.view(-1, 28*28) его размерность станет равна (N, 784).

```
def forward(self, x):
 x = x.view(-1, 28*28)
 x = F.relu(self.fc1(x))
 x = self.fc2(x)
 x = F.softmax(x, dim=1)
 return x
```

2)Для 2 примеры выше, обращаемся по определённым ранее именам слоёв, используется функция relu:

```
 def forward(self, x):
 x = F.relu(self.conv1(x))
 return F.relu(self.conv2(x))
```

3) Пример для сверточной сети

```
 def forward(self, x):
 x = F.max_pool2d(F.relu(self.conv1(x)), (2, 2))
 x = F.max_pool2d(F.relu(self.conv2(x)), 2)
 x = x.view(-1, self.num_flat_features(x))
 x = F.relu(self.fc1(x))
 x = F.relu(self.fc2(x))
 x = self.fc3(x)
 return x
```

4) Пример для свёрточной сети Conv2d -> MaxPool2d -> Conv2d -> MaxPool2d -> Linear -> Linear

```
 def forward(self, input):
 x = self.pool1(F.relu(self.conv1(input)))
 x = self.pool2(F.relu(self.conv2(x)))
 x = x.view(x.size(0), -1)
 x = F.relu(self.fc1(x))
 x = F.relu(self.fc2(x))
 return x
```

4) Пример для рекуррентной сети.

```
 def forward(self, x):
 h0 = Variable(torch.zeros(self.layer_dim, x.size(0), self.hidden_dim))
 out, hn = self.rnn(x, h0)
```

```
 out = self.fc(out[:, -1, :])
 return out
```

Для построения модели надо создать объект описанного класса:

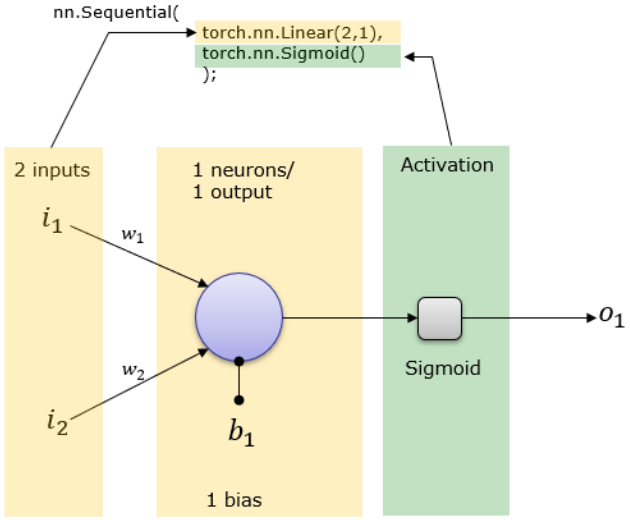
```
Net=Model()
```

На базе *nn.Sequential*. Контейнер для линейных / последовательных слоёв *Linear*, которые имеют вид **W'x+b**, где W — матрица весов размером *(input, output)* и b — вектор смещения размером *output*.

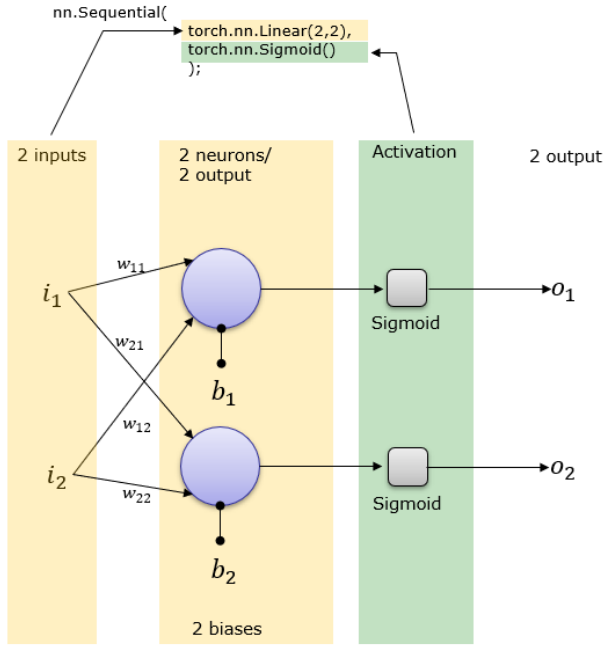
Используя этот контейнер можно в одном операторе определить и вид НС и как будут вычисляться выходные значения.

Пример:

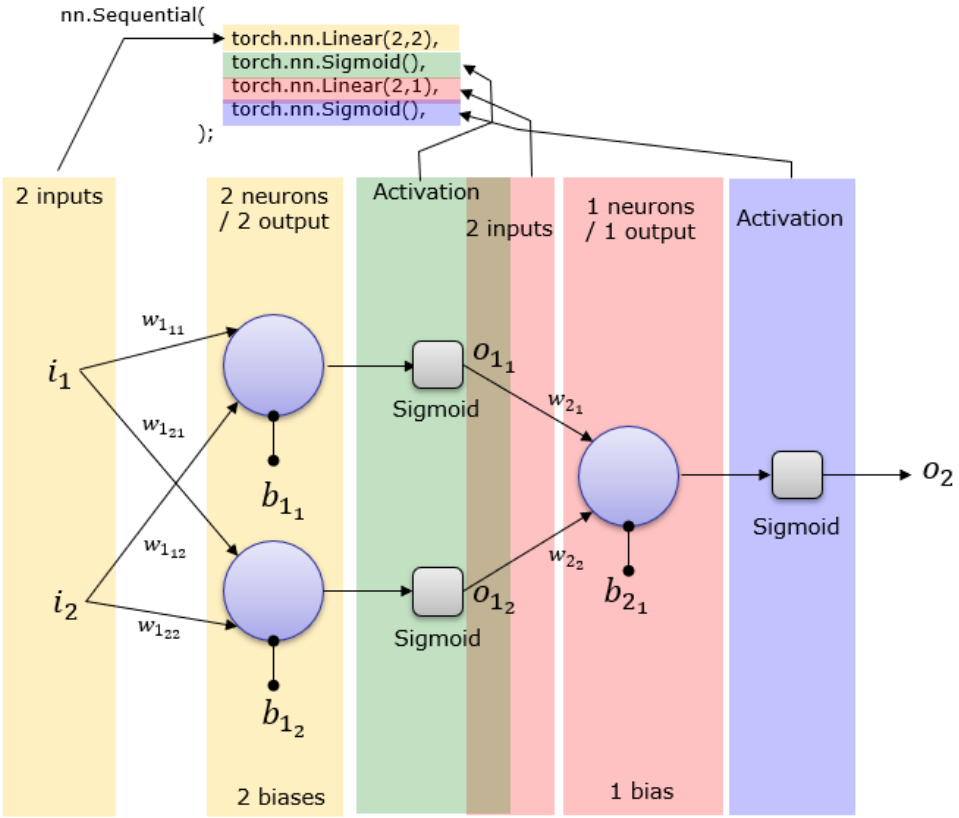
```
НС с 10 входами, с функцией ReLU(), 5 нейронов в скрытом слое, выходной нейрон 1 с функцией сигмойда.
model = nn.Sequential(nn.Linear(10, 5),
 nn.ReLU(),
 nn.Linear(5, 1),
 nn.Sigmoid())
```



а) из 2 входов и 1 выхода



б) 2 входа 2 выхода



в) 2 слоя в первом 2 нейрона во втором 1, функция активации - сигмойда

Рисунок 26 - Примеры простейшего линейного слоя9

[9] http://www.sharetechnote.com/html/Python_PyTorch_nn_Sequential_01.html

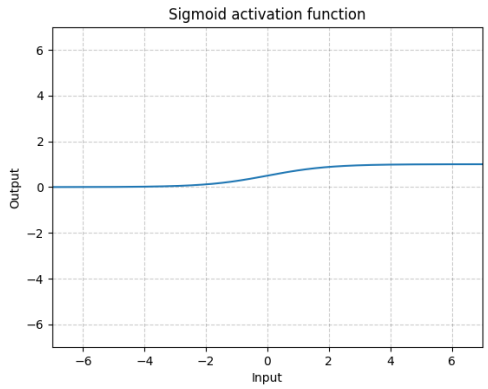
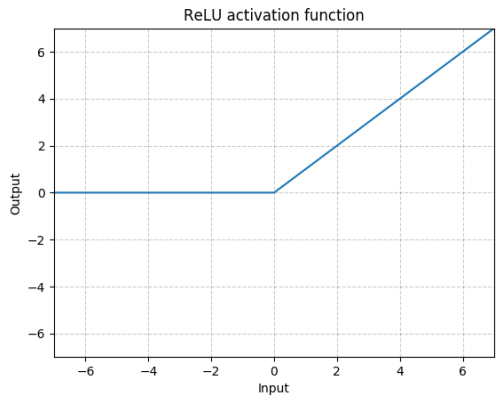
Существуют ещё другие возможности по построению НС. В зависимость от вида НС слои и функции активации могут быть разными., см. подробнее **https://pytorch.org/docs/stable/nn.html**.

Таблица 11 – Примеры типов слоёв НС

Типы слоев	**Назначение**	**Описание**
Активация (activation)	Общие для НС	Содержит функцию активации, которую применяют ко входам слоя.
Нормализация (Normalization)		Слой обеспечивает применение градиентного спуска не к одной точке выборки, а к небольшой коллекции данных.
Прореживание (dropout), регуляризация		Слой обеспечивает добавление информации к условию с целью предотвращения переобучения.
Рекуррентные (Recurrent)	Для рекуррентных НС	Основной блок рекуррентных НС
Свёртка (Convolution)	Для свёрточных НС	основной блок свёрточной нейронной сети, включает в себя для каждого канала свой фильтр, ядро свёртки которого обрабатывает предыдущий слой по фрагментам (суммируя результаты поэлементного произведения для каждого фрагмента).
Пуллинг / группировка / субдискретизация (Pooling)		слои пулинга, как правило, чередуются со слоями свёртки и обобщает (упрощает) информацию, полученную от слоя свёртки, пулинг «сжимает» карты признаков, полученные на предыдущем сверточном слое.
Дополнение отступа (Padding)		пиксели, которые находятся на границе изображения участвуют в меньшем количестве сверток, чем внутренние. В связи с этим в сверточных слоях используется дополнение изображения (англ. padding). Выходы с предыдущего слоя дополняются пикселями так, чтобы после свертки сохранился размер изображения. Такие свертки называют

		одинаковыми (same convolution), а свертки без дополнения изображения называются правильными (valid convolution).
Линейный / Соединение «все-со-всеми» / полносвязный (Linear)	Для НС с прямым распространением и свёрточных	Все входы связаны со всеми нейронами слоя, используются в НС прямого распространения, как последний слой свёрточной сети.

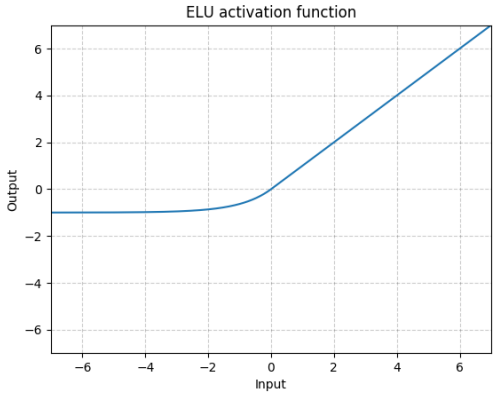
Таблица 12 – Пример функций активации нейронов (подробнее см. https://pytorch.org/docs/stable/nn.html#non-linear-activations-weighted-sum-nonlinearity)

Функция активации	**Метод функции / формула**	**График функции**
`Sigmoid`	`torch.nn.Sigmoid()` $\text{Sigmoid}(x) = \frac{1}{1 + \exp(-x)}$	
`ReLU`	`torch.nn.ReLU`(*inplace=False*) $\text{ReLU}(x) = \max(0, x)$	

ELU

`torch.nn.ELU`(*alpha=1.0, inplace=False*)

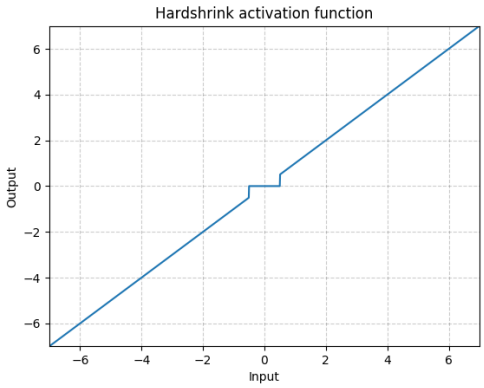
$$\mathrm{ELU}(x) = \max(0, x) + \min(0, \alpha * (\exp(x) - 1))$$



Hardshrink

`torch.nn.Hardshrink`(*lambd=0.5*)

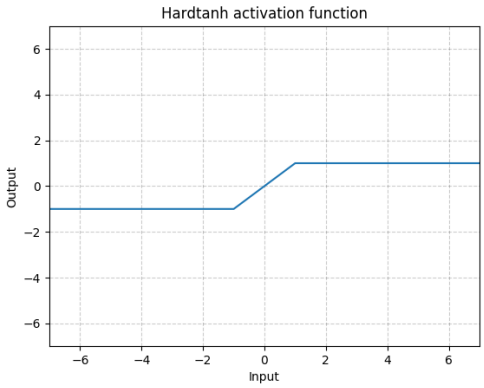
$$\mathrm{HardShrink}(x) = \begin{cases} x, & \text{if } x > \lambda \\ x, & \text{if } x < -\lambda \\ 0, & \text{otherwise} \end{cases}$$



Hardtanh

`torch.nn.Hardtanh`(*min_val=-1.0, max_val=1.0, inplace=False, min_value=None, max_value=None*)

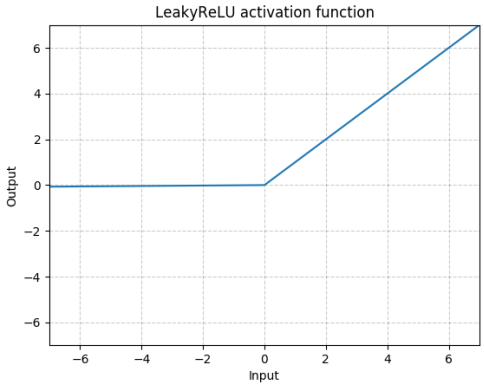
$$\mathrm{HardTanh}(x) = \begin{cases} 1 & \text{if } x > 1 \\ -1 & \text{if } x < -1 \\ x & \text{otherwise} \end{cases}$$



LeakyReLU

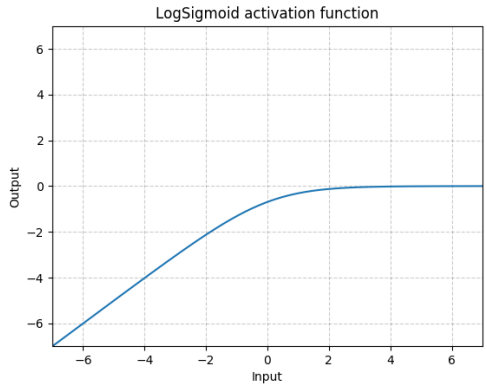
`torch.nn.LeakyReLU`(*negative_slope=0.01, inplace=False*)

$$\mathrm{LeakyRELU}(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \text{negative\_slope} \times x, & \text{otherwise} \end{cases}$$



LogSigmoid	torch.nn.LogSigmoid()

$$\text{LogSigmoid}(x) = \log\left(\frac{1}{1+\exp(-x)}\right)$$



Обучение нейронной сети

Если в качестве обучения используется градиентный спуск (алгоритм обратного распространения ошибки), то процесс обучения выполняется итерационно и включает прямой проход и обратный.

Прямой проход - вычисление выходов НС и текущей ошибки (функции потерь).

Для вычисления выходных значений НС необходимо, подать на входы НС данные из обучающей выборки и последовательно проходить все слои НС.

В библиотеке pytorch в класса Model за выполнение этого действия отвечает метод `forward`, именно в нем задаётся схема вычисления выходов НС.

Пример определения метода `forward` (см. выше определение НС):

```
def forward(self, input):
x = self.pool1(F.relu(self.conv1(input)))
x = self.pool2(F.relu(self.conv2(x)))
x = x.view(x.size(0), -1)
x = F.relu(self.fc1(x))
x = F.relu(self.fc2(x))
return x
```

В PyTorch необязательно вызывать метод в явном виде, при создании объекта НС с указанием входного тензора, метод вызывается. Для вычисления ошибки (функции потерь), необходимо знать текущие значения выходов НС и ожидаемые (те, на которых обучаем). Существует несколько вариантов расчёта потерь (Таблица 13).

Таблица 13 – Примеры функций потерь (подробнее см. https://pytorch.org/docs/stable/nn.html#loss-functions)

Название функции	Синтаксис	Формула
В зависимости от `reduction='none'|'mean'|'sum'`		

$$\ell(x, y) = L = \{l_1, \ldots, l_N\}^\top,$$

$$\ell(x, y) = \begin{cases} \text{mean}(L), \\ \text{sum}(L), \end{cases}$$

Если для поля size_average установлено значение False, потери суммируются для каждой мини-партии. Игнорируется, когда уменьшить является ложным. По умолчанию: True

`L1Loss`	`torch.nn.L1Loss`(*size_average=None, reduce=None, reduction='mean'*)	$l_n = \left\|x_n - y_n\right\|,$
`MSELoss`(средняя квадратичная)	`torch.nn.MSELoss`(*size_average=None, reduce=None, reduction='mean'*)	$l_n = (x_n - y_n)^2$
`KLDivLoss`(информационного расхождения Кульбака-Лейблера)	`torch.nn.KLDivLoss`(*size_average=None, reduce=None, reduction='mean'*)	$l_n = y_n \cdot (\log y_n - x_n)$
`BCELoss`(бинарной кросс-энтропии)	`torch.nn.BCELoss`(*weight=None, size_average=None, reduce=None, reduction='mean'*)	$l_n = -w_n \left[y_n \cdot \log x_n + (1 - y_n) \cdot \log(1 - x_n)\right],$
`BCEWithLogitsLoss`	`torch.nn.BCEWithLogitsLoss`(*weight=None, size_average=None, reduce=None, reduction='mean', pos_weight=None*)	$l_n = -w_n \left[y_n \cdot \log \sigma(x_n) + (1 - y_n) \cdot \log(1 - \sigma(x_n))\right],$
`HingeEmbeddingLoss`	`torch.nn.HingeEmbeddingLoss`(*margin=1.0, size_average=None, reduce=None,*	$l_n = \begin{cases} x_n, & \text{if } y_n = 1, \\ \max\{0, \Delta - x_n\}, & \text{if } y_n = -1, \end{cases}$

reduction='mean')

Пример определения функции потерь:

```
loss_fn = torch.nn.MSELoss(reduction='sum')
```

После того как определено, что НС ещё не обучена (количество итераций меньше заданного числа или функция потерь велика), необходимо обучить НС. Процесс обучения заключается в изменении параметров НС (весов), т.е. выполняется подбор оптимальных значений весов с учётом функции потерь и обучаемой выборки (заданных ожидаемых значений). Для этого используется метод градиентного спуска и реализуется обратный проход.

Веса можно изменять в ручную, изменяя Tensors, содержащие обучаемые параметры (например с помощью torch.no_grad () или .data). Это удобно в случае простых алгоритмов оптимизации, таких как стохастический градиентный спуск, но на практике мы часто обучаем нейронные сети, используя более сложные методы AdaGrad, RMSProp, Adam и т. д. Пакет optim в PyTorch абстрагирует идею алгоритма оптимизации и предоставляет реализации часто используемых алгоритмов оптимизации (Таблица 14).

Таблица 14 – Пример оптимизаторов (подробнее см. https://pytorch.org/docs/stable/optim.html)

Название	**Метод**
SGD	стохастический градиентный спуск
Adam	адаптивная оценка моментов
RMSprop	алгоритм Джеффри Хинтона
LBFGS	алгоритм Бройдена-Флетчера-Гольдфарба-Шанно с ограниченным использованием памяти

Для использования оптимизатора необходмимо на каждой итерации необходимо обнулять градиент, вызывать функцию backward и выполнять шаг оптимизатора.

Пример:

```
optimizer.zero_grad()
loss.backward()
optimizer.step()
```

Использование нейронной сети

После обучения НС ее требуется проверить на тестовой выборки. Если результат удовлетворителен, ее можно использовать для решения поставленной задачи, подавая на вход произвольные значения.

Примеры реализации нейронной сети

1) *Пример НС прямого распространения с функции активации ReLU, случайной выборкой, функцией потерь MSELoss, без использования оптимизатора, задана сеть с помощью nn.Sequential.*

Код:

```
# импорт библиотеки PyTorch
import torch
# задаем значения размерности
N, D_in, H, D_out = 64, 1000, 100, 10
# задаем случайным образом выборки
x = torch.randn(N, D_in)
y = torch.randn(N, D_out)
# строим модель НС
model = torch.nn.Sequential(
    torch.nn.Linear(D_in, H),
    torch.nn.ReLU(),
    torch.nn.Linear(H, D_out),
)
# выбрали функцию потерь
loss_fn = torch.nn.MSELoss(reduction='sum')
# скорость обучения
learning_rate = 1e-4
# цикл обучения с 500 эпохами
for t in range(500):
    y_pred = model(x)
    loss = loss_fn(y_pred, y)
    if t % 100 == 99:
        print(t, loss.item())
    model.zero_grad()
    loss.backward()
# расчет вручную параметрой НС
    with torch.no_grad():
        for param in model.parameters():
            param -= learning_rate * param.grad
```

2) *Пример НС прямого распространения со случайной выборкой с 2 слоями, первый с функции активации ReLU, выходной с сигмоидальной функцией, используется для обучения метод обратного распространения и стохастический градиентный*

спуск (SGD), в качестве функции потерь средняя квадратичная функция (MSELoss).

Код:

```
# импорт библиотеки PyTorch
import torch
import torch.nn as nn

# определить все слои и размер пакета
# n_in -входной, n_h - скрытый,  n_out - выходной, batch_size - пакет
обучающей выборки
n_in, n_h, n_out, batch_size = 10, 5, 1, 9

# заполняем  случайными числами
#Возвращает тензор, заполненный случайными числами из нормального
распределения со средним 0 и дисперсией 1
#(также называемый стандартным нормальным распределением).Форма тензора
определяется переменным размером аргумента.

# входные данные
x = torch.randn(batch_size, n_in)
print(x)

#Создает тензор с данными.
#выходные данные
y = torch.tensor([[1.0], [0.0], [0.0], [1.0], [1.0], [1.0], [0.0], [0.0],
[1.0]])
print(y)

#class torch.nn.Sequential(*args) - Последовательный контейнер (модель НС).
#Модули будут добавлены к нему в порядке их передачи в конструктор.
# Линейное преобразование входных данных y=x*(A)T+b
# определяем слой входов 10, слой с функцией ReLU() содержит 5 нейронов,
выходной нейрон 1 с функцией сигмойда
model = nn.Sequential(nn.Linear(n_in, n_h),
   nn.ReLU(),
   nn.Linear(n_h, n_out),
   nn.Sigmoid())

# выбрали функцию потерь MSELoss()
criterion = torch.nn.MSELoss()
# выбрали метод оптимизации и установили скорость обучения
optimizer = torch.optim.SGD(model.parameters(), lr = 0.01)
# модель градиентного спуска
```

```
# цикл обучения
for epoch in range(100):
   # Прямой проход: вычисляем выход НС, подав на вход модели начальные
значения X
   y_pred = model(x)

   print(y_pred)
   # расчитываем функцию потерь
   loss = criterion(y_pred, y)
   print('epoch: ', epoch,' loss: ', loss.item())

   # Нулевые градиенты, выполнить обратный проход и обновить веса.
   # В PyTorch нам нужно установить градиенты на ноль, прежде чем начинать
обратное распространение,
   # поскольку PyTorch накапливает градиенты при последующих обратных
проходах.
   optimizer.zero_grad()

   # обратный проход (вычисляются градиенты)
   loss.backward()

   # шаг спуска градиента
   optimizer.step()

# новый входной вектор
x1 = torch.randn(1, n_in)
print(x1)

# получение выхода обученной НС
y_pred2 = model(x1)
print(y_pred2)
```

3) *Пример НС (Linear –> ReLU –> Linear), определённой с помощью* nn.Module*, с использование оптимизатора Adam, с функцией потерь CrossEntropyLoss, обучающая выборка набор MNIST .*

Код:

```
import torch
import torch.nn as nn
import torchvision.datasets as dsets
import torchvision.transforms as transforms
from torch.autograd import Variable
# Размеры изображения = 28 x 28 = 784
```

```
input_size = 784
# Количество узлов на скрытом слое
hidden_size = 500
# Число классов на выходе. В этом случае от 0 до 9
num_classes = 10
# Количество тренировок всего набора данных
num_epochs = 5
# Размер входных данных для одной итерации
batch_size = 100
# Скорость обучения
learning_rate = 0.001
# Грузим набор данных MNIST
# обучающая выборка
train_dataset = dsets.MNIST(
    root='./data',
    train=True,
    transform=transforms.ToTensor(),
    download=True
)
# тестовая выборка
test_dataset = dsets.MNIST(
    root='./data',
    train=False,
    transform=transforms.ToTensor()
)
# создаем загрузчик для НС с перемешиванием для обучающей выборки и без
перемешивания для тестовой.
train_loader = torch.utils.data.DataLoader(
    dataset=train_dataset,
    batch_size=batch_size,
    shuffle=True
)
test_loader = torch.utils.data.DataLoader(
    dataset=test_dataset,
    batch_size=batch_size,
    shuffle=False
))
# Определяем вид НС
class Net(nn.Module):
    def __init__(self, input_size, hidden_size, num_classes):
        super(Net, self).__init__()                    # Наследуемый
родительским классом nn.Module
```

```
        self.fc1 = nn.Linear(input_size, hidden_size)  # 1й связанный слой:
784 (данные входа) -> 500 (скрытый узел)
        self.relu = nn.ReLU()                          # Нелинейный слой ReLU
max(0,x)
        self.fc2 = nn.Linear(hidden_size, num_classes) # 2й связанный слой:
500 (скрытый узел) -> 10 (класс вывода)

    def forward(self, x):                              # Прямой проход
        out = self.fc1(x)
        out = self.relu(out)
        out = self.fc2(out)
        return out
# создаем объект НС
net = Net(input_size, hidden_size, num_classes)
# Определяем функцию потерь
criterion = nn.CrossEntropyLoss()
# Определяем оптимизатор
optimizer = torch.optim.Adam(net.parameters(), lr=learning_rate)
#Цикл обучения
for epoch in range(num_epochs):
# Грузим пакет изображений (index, data, class)
    for i, (images, labels) in enumerate(train_loader):
        # Преобразуем тензор в Variable: вектор  784 в матрицу 28 х 28
        images = Variable(images.view(-1, 28*28))
        labels = Variable(labels)
        # Инициализируем скрытые слои, веса=0
        optimizer.zero_grad()
        # Прямой проход по НС; вычисляем выход
        outputs = net(images)
        # Вычисляем функцию потерь (ошибку)
        loss = criterion(outputs, labels)
        # Обратный проход: корректировка весов
        loss.backward()
        # Выполняем шаг оптимизации (обучения)
        optimizer.step()
        # Выводим данные по обучению
        if (i+1) % 100 == 0:
            print('Epoch [%d/%d], Step [%d/%d], Loss: %.4f'%(epoch+1,
num_epochs, i+1, len(train_dataset)//batch_size, loss))
correct = 0
total = 0
for images, labels in test_loader:
    images = Variable(images.view(-1, 28*28))
```

```
    outputs = net(images)
    _, predicted = torch.max(outputs.data, 1)  # Выбор лучшего класса из
выходных данных: класс с лучшим счетом
    total += labels.size(0)                    # Увеличиваем суммарный счёт
    correct += (predicted == labels).sum()     # Увеличиваем корректный счёт
print('Точность сети на 10К тестовых изображений: %d %%' % (100 * correct /
total))
torch.save(net.state_dict(), 'fnn_model.pkl')
```

4) *Пример Сверточной сети НС (Conv2d -> MaxPool2d -> Conv2d -> MaxPool2d -> Linear -> Linear), определённой с помощью nn.Module, с функцией потерь CrossEntropyLoss, случайной обучающей выборкой.*

Код:

```
import torch
from torch.autograd import Variable
import torch.nn as nn
import torch.nn.functional as F
class MNISTConvNet(nn.Module):
    def __init__(self):
        super(MNISTConvNet, self).__init__()
        self.conv1 = nn.Conv2d(1, 10, 5)
        self.pool1 = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(10, 20, 5)
        self.pool2 = nn.MaxPool2d(2, 2)
        self.fc1 = nn.Linear(320, 50)
        self.fc2 = nn.Linear(50, 10)
    def forward(self, input):
        x = self.pool1(F.relu(self.conv1(input)))
        x = self.pool2(F.relu(self.conv2(x)))
        x = x.view(x.size(0), -1)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        return x
net = MNISTConvNet()
print(net)
input = Variable(torch.randn(1, 1, 28, 28))
out = net(input)
print(out.size())
target = Variable(torch.LongTensor([3]))
loss_fn = nn.CrossEntropyLoss()
err = loss_fn(out, target)
err.backward()
print(err)
```

```
print(net.conv1.weight.data.norm())
print(net.conv1.weight.grad.data.norm())
```

5) *Пример рекуррентной сети с функцией relu обучаемой на входных данных набора MNIST, оптимизатором SGD и функцией потерь CrossEntropyLoss.*

```
#https://www.deeplearningwizard.com/deep_learning/practical_pytorch/pytorch_r
ecurrent_neuralnetwork/
# подключаем библиотеки
import torch
import torch.nn as nn
import torchvision.transforms as transforms
import torchvision.datasets as dsets

# создаем обучающую и тестовую выборки на базе  MNIST
train_dataset = dsets.MNIST(root='./data',
                            train=True,
                            transform=transforms.ToTensor(),
                            download=True)
test_dataset = dsets.MNIST(root='./data',
                           train=False,
                           transform=transforms.ToTensor())
# задаем параметры
batch_size = 100
n_iters = 3000
num_epochs = n_iters / (len(train_dataset) / batch_size)
num_epochs = int(num_epochs)
# подаем данные в загрузчик
train_loader = torch.utils.data.DataLoader(dataset=train_dataset,
                                           batch_size=batch_size,
                                           shuffle=True)
test_loader = torch.utils.data.DataLoader(dataset=test_dataset,
                                          batch_size=batch_size,
                                          shuffle=False)
# определяем рекуррентную нейронную сеть
class RNNModel(nn.Module):
    def __init__(self, input_dim, hidden_dim, layer_dim, output_dim):
        super(RNNModel, self).__init__()
        # размерность скрытых слоев]
        self.hidden_dim = hidden_dim
        # Количество скрытых слоев
        self.layer_dim = layer_dim
```

```
        self.rnn = nn.RNN(input_dim, hidden_dim, layer_dim, batch_first=True,
nonlinearity='relu')
        # Слой считывания
        self.fc = nn.Linear(hidden_dim, output_dim)

    def forward(self, x):
        h0 = torch.zeros(self.layer_dim, x.size(0),
self.hidden_dim).requires_grad_()
        out, hn = self.rnn(x, h0.detach())
        # out.size() --> 100, 28, 10
        out = self.fc(out[:, -1, :])
        # out.size() --> 100, 10
        return out
# задаем параметры НС
input_dim = 28
hidden_dim = 100
layer_dim = 1
output_dim = 10
learning_rate = 0.01
criterion = nn.CrossEntropyLoss()
model = RNNModel(input_dim, hidden_dim, layer_dim, output_dim)
optimizer = torch.optim.SGD(model.parameters(), lr=learning_rate)
seq_dim = 28
iter = 0
for epoch in range(num_epochs):
    for i, (images, labels) in enumerate(train_loader):
        model.train()
        # Загрузка изображений в виде тензоров с возможностью накопления
градиента
        images = images.view(-1, seq_dim, input_dim).requires_grad_()
        # Обнуление градиента
        optimizer.zero_grad()
```

Варианты для выполнения лабораторной работы

Вариант	Функция потерь	Функция активации	НС	Оптимизатор	Входные данные
1	L1Loss	Sigmoid	Свёрочная НС (не менее 2 свёрточных слоёв и одного линейного)	RMSProp	**MNIST**
2	MSELoss	ReLU	НС прямого распространения (использовать не менее 2 разных нелинейных функций)	SGD	CIFAR10

3	KLDivLoss	ELU	Рекуррентная нейронная сеть с 2 нелинейными слоями (readout)	Adam	STL10
4	BCELoss	Hardshrink	НС прямого распространения (использовать не менее 2 разных нелинейных функций)	Adagrad	Рандом
5	BCEWithLogitsLoss	Hardtanh	Рекуррентная нейронная сеть с 2 нелинейными слоями (readout)	SGD	CIFAR10
6	HingeEmbeddingLoss	LeakyReLU	Свёрочная НС (не менее 2 свёрточных слоёв и одного линейного)	Adam	STL10
7	KLDivLoss	LogSigmoid	Свёрочная НС (не менее 2 свёрточных слоёв и одного линейного)	RMSProp	CIFAR10
8	BCELoss	ELU	НС прямого распространения (использовать не менее 2 разных нелинейных функций)	SGD	STL10
9	BCEWithLogitsLoss	Hardshrink	Рекуррентная нейронная сеть с 2 нелинейными слоями (readout)	Adam	**MNIST**
10	L1Loss	Hardtanh	НС прямого распространения (использовать не менее 2 разных нелинейных функций)	RMSProp	CIFAR10
11	MSELoss	LeakyReLU	Рекуррентная нейронная сеть с 2 нелинейными слоями (readout)	SGD	STL10
12	KLDivLoss	LogSigmoid	Свёрочная НС (не менее 2 свёрточных слоёв и одного линейного)	Adam	Рандом
13	BCELoss	Sigmoid	Рекуррентная нейронная сеть с 2 нелинейными слоями (readout)	Adagrad	CIFAR10
14	KLDivLoss	LogSigmoid	Свёрочная НС (не менее 2 свёрточных слоёв и одного линейного)	RMSProp	CIFAR10
15	BCELoss	ELU	Рекуррентная нейронная сеть с 2 нелинейными слоями (readout)	SGD	STL10
16	BCEWithLogitsLoss	Hardshrink	НС прямого распространения (использовать не менее 2 разных нелинейных функций)	Adam	CIFAR10
17		Hardtanh	НС прямого распространения (использовать не менее 2 разных нелинейных функций)	RMSProp	STL10

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ

Список рекомендуемой литературы

основная

1) Смагин, А. А. Интеллектуальные информационные системы : учеб. пособие для вузов / А. А. Смагин, С. В. Липатова, А. С. Мельниченко ; УлГУ, Фак. математики и информ. технологий, Каф. телекоммуникац. технологий и сетей. - Ульяновск : УлГУ, 2010.- URL: ftp://10.2.96.134/Text/smagin2.pdf
2) Станкевич, Л. А. Интеллектуальные системы и технологии : учебник и практикум для бакалавриата и магистратуры / Л. А. Станкевич. — Москва : Издательство Юрайт, 2019. — 397 с. — (Бакалавр и магистр. Академический курс). — ISBN 978-5-534-02126-4. — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://www.biblio-online.ru/bcode/433370

дополнительная

3) Каку М., Будущее разума [Электронный ресурс] / Каку М. - М. : Альпина Паблишер, 2016. - 502 с. - ISBN 978-5-91671-369-5 - Режим доступа: http://www.studentlibrary.ru/book/ISBN9785916713695.html
4) Потопахин В.В., Романтика искусственного интеллекта / Потопахин В. В. - М. : ДМК Пресс, 2017. - 170 с. - ISBN 978-5-97060-476-2 - Текст : электронный // ЭБС "Консультант студента" : [сайт]. - URL : https://www.studentlibrary.ru/book/ISBN9785970604762.html (дата обращения: 10.11.2020). - Режим доступа : по подписке.
5) Цуканова Н.И., Онтологическая модель представления и организации знаний : Учебное пособие для вузов / Цуканова Н.И. - М. : Горячая линия - Телеком, 2015. - 272 с. - ISBN 978-5-9912-0454-5 - Текст : электронный // ЭБС "Консультант студента": [сайт]. - URL: https://www.studentlibrary.ru/book/ISBN9785991204545.html (дата обращения: 10.11.2020). - Режим доступа : по подписке.
6) Седова, Н. А. Теория нечетких множеств : учебное пособие / Н. А. Седова, В. А. Седов. — Саратов : Ай Пи Ар Медиа, 2019. — 421 с. — ISBN 978-5-4497-0196-1. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/86526.html

7) Павлова, А. И. Информационные технологии: основные положения теории искусственных нейронных сетей : учебное пособие / А. И. Павлова. — Новосибирск : Новосибирский государственный университет экономики и управления «НИНХ», 2017. — 191 с. — ISBN 978-5-7014-0801-0. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/87110.html

учебно-методическая

8) Седов, В. А. Введение в нейронные сети : методические указания к лабораторным работам по дисциплине «Нейроинформатика» для студентов специальности 09.03.02 «Информационные системы и технологии» / В. А. Седов, Н. А. Седова. — Саратов : Ай Пи Эр Медиа, 2018. — 30 с. — ISBN 978-5-4486-0047-0. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/69319.html

Программное обеспечение

Anaconda (дистрибутив языков программирования Python и R), библиотеки (open source).