ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет информатики, математики и компьютерных наук

Направление

«Интеллектуальный анализ данных»

ПРОЕКТ ПО НАУЧНО-ИССЛЕДОВАТЕЛЬСКОМУ СЕМИНАРУ

На тему «Алгоритмы анализа данных в системе визуализации запахов»

Анна Белова

Роман Власов

Александр Лапшин

Андрей Латышев

Дмитрий Семенов

Андрей Тимофеев

Лидия Торопова

Lead: Алина Шадрина

Нижний Новгород, 2016

Оглавление

ВВЕДЕНИЕ

Предметом исследования настоящего проекта являются алгоритмы обработки и визуализации данных в системе искусственного обоняния. Объект исследования — данные, полученные в результате химических экспериментов по разработке системы «Электронный нос».

Основной целью проекта была разработка прототипа системы анализа и визуализации данных в системе искусственного обоняния. Для достижения поставленной цели были поставлены следующие задачи:

1. Изучение системы электронный нос и поставляемых данных
2. Разработка алгоритмов обработки данных для электронного носа
3. Разработка прототипа системы обработки данных
4. Разработка архитектуры системы визуализации запахов на основе технологии смешанной реальности
5. Разработка прототипов модулей системы визуализации e-dog.

В результате, ожидается получить:

1) Алгоритм N-label классификации, обученный на распознавание как отдельных органических веществ, так и их смесей, включенных в тренировочный набор, а затем классифицировать тестовый набор, состоящий из запахов игрушек.
2) прототип системы визуализации запахов на основе информации о классах опасности веществ.

Исследование выполнено на языке Python с использованием библиотеки sklearn, модулей xlread, transliterate, NumPy, SciPy, matplotlib

Результаты исследования и исходный код находятся по ссылке: https://github.com/ashadrina/nose_project

1. ИССЛЕДОВАНИЕ ДАННЫХ И ОБУЧЕНИЕ С УЧИТЕЛЕМ

Выполнила Алина Шадрина

1.1 Данные в XLS

Для начального исследования и предобработки получены 33 объекта тренировочной выборки, которые представляют собой отдельные вещества, 4 объекта валидационной – смеси двух веществ, и 75 новых объектов, которые необходимо классифицировать. Каждый объект хранится в файле вида название_вещества.XLS и представляет собой таблицу, содержащую следующие блоки:

1. Шапка: название (вещества или игрушки, или состав смеси), продолжительность (всегда 120 с), тип (обычно значение «измерение», назначение этого поля не исследовалось), статистические данные (обычно значение «нет», назначение этого поля не исследовалось), начало (число-время начала измерения).
2. Информация о сенсорах: 8 пар вида «название сенсора – базовая частота сенсора».
3. Матрица 120 x 8, где столбцы соответствуют сенсорам, а строки – временным отсчетам. Таким образом, каждый элемент матрицы отражает изменение частоты сенсора i (i=[1,8]) в момент времени j (j=[0,120])

Для обучения получены следующие вещества (см.рисунок 1):

- диоктилфталат – 9 шт. в разных концентрациях на разных носителях;
- ацетальдегид, ацетон, бензол, этилацетат - 4 шт. в разных концентрациях;
- пластизоль – 2 шт.;
- бензин, бутанол, бутилацетат, гексан, изобутанол, изопропанол, пропанол, стирол, толуол, фенол – 1шт.;

Метки классов извлекаются автоматически из названий файлов. Правило именования файлов выглядит следующим образом: «название_вещества

[концентрация] мкл на [носитель]» (носитель и концентрация опциональны). Для формирования датасета было решено не делать различий между одним и тем же веществом в разной концентрации или на разных носителях, поэтому алгоритм извлечения меток классов состоит в том, чтобы разрезать название файла по пробелам и сохранять первый (в индексах списков Python - нулевой) элемент.

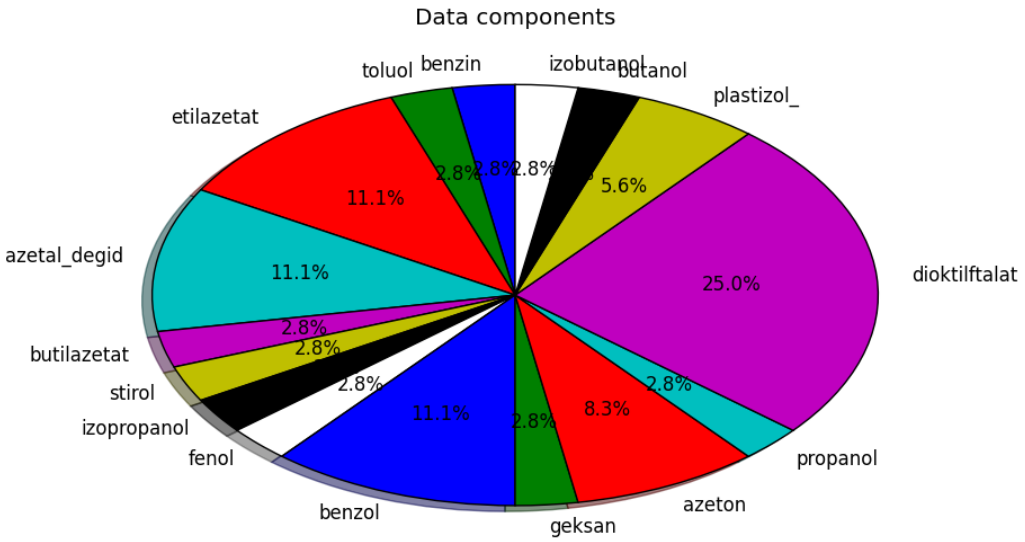


Рисунок 1 – Состав тренировочного множества

В процессе исследования тренировочных данных выявлены следующие проблемы и предложены пути их решения:

1. **Проблема:** опечатки в названия файлов

   На что влияет: на автоматическое извлечение меток классов

   Решение: переименовать файл вручную
2. **Проблема:** кириллица в названиях файлов

   На что влияет: бинаризация меток классов не работает с кириллицей => обучение с учителем невозможно

   Решение: выполнить примитивную транслитерацию (см. файл xls_parser.py, метод cyrillic2latin)
3. **Проблема:** Не все файлы содержат матрицу подходящего размера

   На что влияет: невозможно сформировать датасет

Решение: включит проверку при считывании матриц (см. файл xls_parser.py, метод main)

4. **Проблема:** датасет слишком маленький и несбалансированный

   На что влияет: невозможно качественное обучение с учителем

5. **Решение:** а) работа с базой данных sniff (см. п.1.2) и б) генерация искусственных данных 8 (см. п. 1.4)

В процессе работы над данными изменился подход к формату датасета, поэтому в репозиторий содержит более старые данные (test_data, train_data, train_labels). В частности, в новом датасете смеси вынесены в отдельный файл, изменились названия датасетов, а также изменился формат файлов на более удобный для дальнейшего парсинга: теперь столбцы матриц разделены прямым слешем «|», а отсчеты внутри столбцов – точкой с запятой «;».

Результатом данного шага стали:

1. Код:

   https://github.com/ashadrina/nose_project/blob/master/code/xls_parser.py

   Запускать: `python xls_parser.py train train.txt`

   где train ИЛИ test ИЛИ val – каталоги, содержание XLS файлы с данными, txt файл – имя выходного файла

2. 3 датасета:

   a. Тренировочный – вещества, 120 x 33

   https://github.com/ashadrina/nose_project/blob/master/data/data_train.txt

   b. Валидационный – смеси – 120 x 4

   https://github.com/ashadrina/nose_project/blob/master/data/data_val.txt

   c. Тестовый – игрушки – 120 x 75

   https://github.com/ashadrina/nose_project/blob/master/data/data_test.txt

2. Вывод о необходимости расширения датасета

1.2 Данные в БД

Выполнила Лидия Торопова

Предполагалось, что из базы данных sniffdb.sdf удастся извлечь дополнительные данные и таким образом расширить множества веществ и

смесей. Файл базы данных создан в Microfost SQL Server Compact Edition, что само по себе является серьезным недостатком, так как в команде были ноутбуки не только под ОС Windows, но также под OS X и Linux. Кроме того, это устаревший формат, несовместимый с прочими инструментами Microsoft для работы с базами данных. Для обеспечения кросс-платформенного доступа база с помощью инструмента SDF Viewer была конвертирована в sql-файл, который, в свою очередь был скорректирован для работы с mysql5.5.

База данных содержит:

- 8 таблиц: Data, GroupTree, Mask, MaskData, MeasureProfile, MeasureProfileData, Measures, Sensors;
- 430 записей измерений и прочистки сенсоров;
- 250 различных объектов;
- 103 объекта длины 120 (объекты были отфильтрованы по полю FullLength как по самому критичному, дальнейший анализ извлеченных измерений предполагалось провести вручную);
- Огромное количество мусора.

Для автоматизации работы с БД был написан скрипт на Python parse_sql.py. В процессе анализа извлекаемой информации было обнаружено, что в таблице Data хранятся не изменения частот сенсоров, а значения частот. Вычисление необходимых матриц ΔFпоказало, что частоты изменяются «ступенькой», что отличается от уже имеющихся данных из XLS, где они изменяются плавно. Возможная причина состоит в том, что система «электронный нос» совершенствовалась, поэтому данные из БД сделаны более старой версией анализатора, а данные в XLS - более новые. Таким образом, было принято решение отказаться от дальнейшей работы с этой базой.

Результаты данного шага:

1. SQL-файл https://github.com/ashadrina/nose_project/blob/master/data/sniffdb_export_120.sql
2. Код для работы с Mysql

https://github.com/ashadrina/nose_project/blob/master/code/mysql_parse.py

Запускать: из sniffdb_export_120.sql создать БД в mysql,

```
python mysql_parse.py out.txt
```

3. Необходимость искать другие способы наполнения датасета

1.3 Исследование исходных данных

В качестве начального шага было проведено исследование данных методов сингулярного разложения. На рисунке 2 видно, что во всех трёх датасетах основную информацию несёт только 1 компонента. Остальные можно игнориоровать, таким образом превратив датасет в множество векторов, а не в множество матриц.

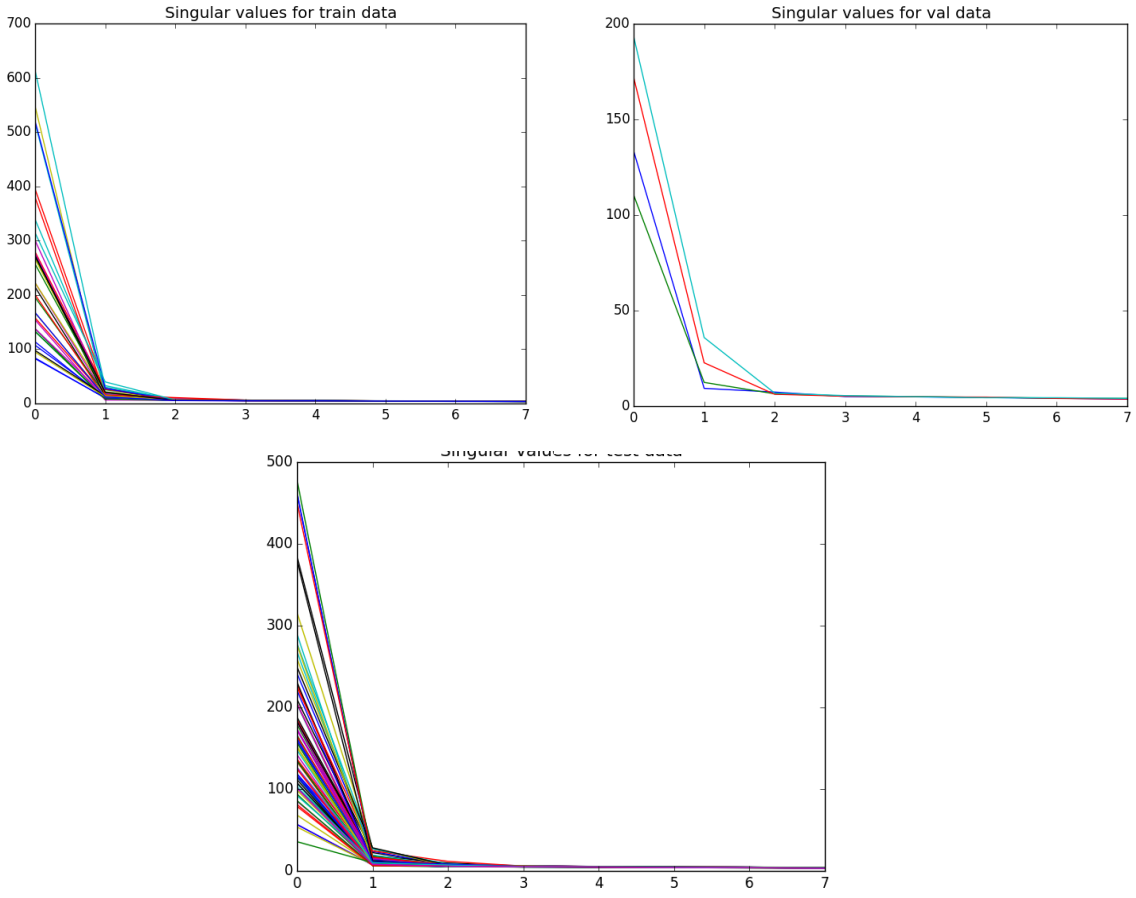


Рисунок 2 – Графики сингулярных чисел для тренировочного, валидационного и тестового датасетов

Данные графики в полном размере можно посмотреть на репозитории по ссылкам:

https://github.com/ashadrina/nose_project/blob/master/code/train_svd.png

https://github.com/ashadrina/nose_project/blob/master/code/val_svd.png

https://github.com/ashadrina/nose_project/blob/master/code/test_svd.png

1.4 Генерация искусственных данных и обучение с учителем

Генерация искусственных данных преследовала 2 цели:

1. Получить б**о**льшее тренировочное множество
2. Сбалансировать классы

В данной работе реализован простейший алгоритм, вдохновлённый генерацией искусственных данных в обработке изображений: нужно просто добавлять константу из эталонных данных. В качестве альтернативных вариантов рассматривались остальные арифметические операции.

Кроме того, подбирался следующий параметр – количество генерируемых данных каждого класса. Чтобы удовлетворить требованию сбалансированности, это число рассчитывалось следующим образом:

$$N\_new = N\_exp - N\_real,$$

где N_new – количество данных одного класса, которые необходимо сгенерировать, N_exp – количество данных одного класса, которое мы хотели бы получить, и N_real – количество «настоящих» данных в датасете.

Оба параметра – арифметическая операция и количество новых данных – подбирались по результатам обучения нескольких классификаторов. Описанные методы не являются лучшими, но в условиях ограниченности времени они работают.

Способ генерации смесей не найден, так как мы не располагаем достаточным для анализа количеством смесей. Однако, этот способ значительно повысил бы точность классификации новых данных.

Перейдём непосредственно к задаче классификации и обучению. Для подбора подходящего метода генерации данных в обучении и тестировании использовались только данные тренировочного датасета. В таблице 1 приведены результаты подбора наилучшего метода – добиваемся присутствия 10 объектов каждого класса с помощью операций прибавления или вычитания констант.

№	Данные	SVM	Knn	Rand.Forest	GussianNB
1	Начальные данные	20% / 36%	48% / 0%	100% / 9%	76% / 9%
2	Train – начальные, Test - сгеренированные	2% / 0%	5% / 0%	100% / 0%	61% / 0%
3	Cross validation, 10 (+,-)	18% / 20%	**92% / 81%**	100% / 64%	20% / 22%
4	Cross validation, 20 (+,-)	15% / 17%	87% / 78%	100% / 62%	20% / 20%
5	Cross validation, 10 (+,-,*,\)	13% / 7%	84% / 74%	97% / 66%	6% / 7%

Таблица 1 – Результаты подбора метода генерации данных

Затем, когда был выбран наилучший метод генерации, он был протестирован на смесях.Результаты приведены в таблице 2. Наилучший результат показал наивный байесовский классификатор – при обучении на смеси настоящих и искусственных веществах 83% и при тестировании на смесях точность 82%.

Данные	SVM	Knn	Rand.Forest	GussianNB
Train – начальные + сгенерированные методом 10 (+,-), Test - смеси веществ	**83% / 87%**	98% / 81%	100% / 81%	83% / 82%

Таблица 2 – Результаты тестирования метода генерации данных

Последним шагом в тренировочное множество были объединены начальные вещества, данные, сгенерированные методом 10 (+,-), и смеси и проведена классификация новых данных. Так как нет возможности проверить точность классификации, было принято решение задействовать больше алгоритмов и считать верным тот результат, который показало наибольшее число алгоритмов. Поскольку таблица большая, полностью её можно скачать с гитхаб:

https://github.com/ashadrina/nose_project/blob/master/classification_report.xlsx

Результаты данного шага:

1. Код, который генерирует данные и запускает обучение с учителем

   https://github.com/ashadrina/nose_project/blob/master/code/generate_data.py

   Запускать финальную классификацию: `python generate_data.py`
2. Файл с результатами экспериментов:

https://github.com/ashadrina/nose_project/blob/master/code/training_out.txt

https://github.com/ashadrina/nose_project/blob/master/code/test_output.txt

3. Результат классификации новых данных:

https://github.com/ashadrina/nose_project/blob/master/classification_report.xlsx

2. АЛЬТЕРНАТИВНЫЙ МЕТОД ГЕНЕРАЦИИ ДАННЫХ

Выполнил Дмитрий Семёнов

Помимо описанного выше алгоритма генерации данных, был предложен альтернативный вариант. Так как входные данные представлены матрицами 8x120, то можно взять не все 120 фичей, а лишь часть из них. То есть предложенный метод – взять, например, 70 фич из 120 и перебрать все сочетания из 120 по 70 (это очень большое число) – это будут различные представления одного и того же вещества. Было решено взять число фич, близкое к 120, чтобы искусственные данные максимально отражали действительность. Для теста все сочетания из 118 фич. Получим (118*119)/2 представлений каждого вещества. Соответственно для них проведем тест данного метода на смесях.

Данные	SVM	Knn	Rand.Forest
Train – начальные + сгенерированные, Test - смеси веществ	**65%**	61%	70%

Как мы видим, результаты очень даже неплохие, но ошибка все равно составляет около 30% в лучшем случае. Но, возможно, при большем количестве данных можно будет генерировать по 119 фичам.

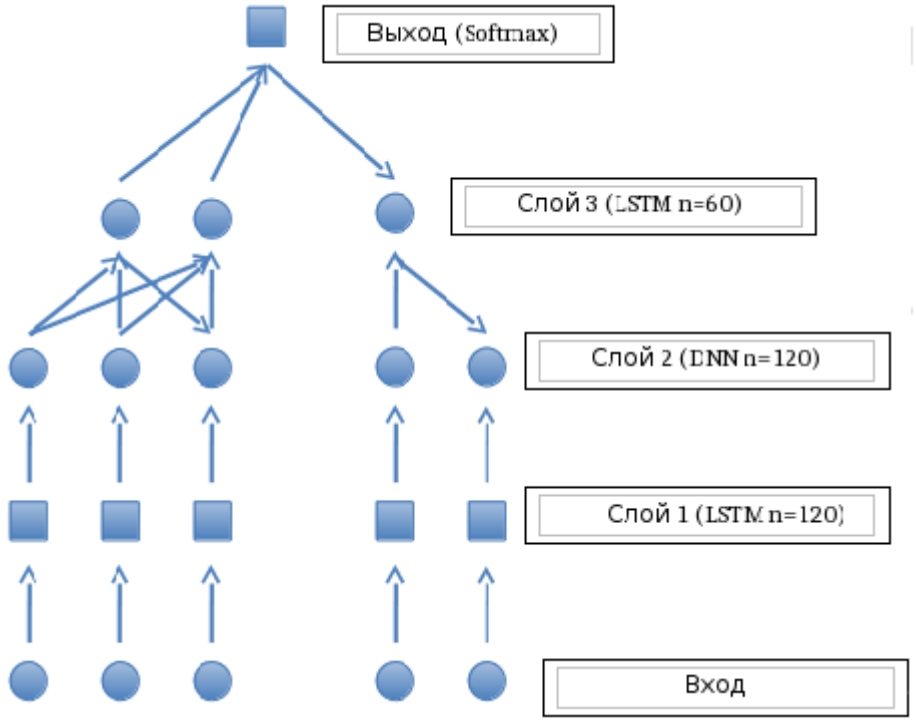
Также важно заметить, что этот эксперимент проводился до определения главных компонент. То есть имеет смысл провести испытания для сгенерированных данных, где обязательно присутствует главная компонента. Процент ошибок получился более низким:

Данные	SVM	Knn	Rand.Forest
Train – начальные + сгенерированные, Test - смеси веществ	**73%**	64%	78%

3. НЕЙРОННЫЕ СЕТИ

Выполнили Роман Власов и Андрей Тимофеев

Бала реализована рекуррентная нейронная сеть (LSTM) с помощью библиотеки tensorflow и ее высокоуровневого интерфейса skflow на языке программирования Python. Данная сеть имеет динамическую структуру, т.е. количество слоев и количество нейронов на каждом слое можно изменять для подбора наилучшей архитектуры. Так же есть возможность применить dropout, установить размер mini-batch и изменить темп обучения.



Были протестированные некоторые конфигурации нейронной сети на сгенерированных данных, которые включали в себя: применения dropout, увеличение слоев, изменение количества нейронов на каждом слое, использование mini-batch, разные значения параметра темпа обучения.

В ходе данного исследования было выявлено, что нейронная сеть со следующей структурой (темп обучения = 0.03) показала наилучший результат 62.2312%

Данный результат намного ниже чем у классических алгоритмов машинного обучения. Вероятно, это вызвано тем, что сеть недоучилась на сгенерированных данных.

Ссылка на гитхаб:
https://github.com/ashadrina/nose_project/blob/master/code/lstm.py

4. АНАЛИЗ НА ОСНОВЕ КЛАССОВ ОПАСНОСТИ

Выполнил Латышев Андрей

С точки зрения практического применения полезно уметь определять, какие вещества могут нанести вред жизни и здоровью человека, а какие безвредны. Существует классификация химических веществ на 5 классов опасности (подробнее: https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_%D0%BE%D0%BF%D0%B0%D1%81%D0%BD%D0%BE%D1%81%D1%82%D0%B8)

Самыми опасными, согласно данной классификации признаются вещества, относящиеся к 1-му классу, а безвредными – вещества 5-го класса опасности.

Для веществ из тренировочного множества классификация выглядит таким образом:

- Класс опасности 5: диоктилфталат.
- Класс опасности 4: бутилацетат, пластизоль, ацетон.
- Класс опасности 3: бутанол, ацетальдегид, изобутанол, пропанол, этилацетат, бензин.
- Класс опасности 2: бензол, фенол.
- Класс опасности 1: нет представителей.

Однако научиться предсказывать класс опасности по показаниям датчиков не удалось т.к. кластеры оказались слабо разделимы. Отсюда получаем вывод о том, что система «Электронный нос» не позволяет с приемлемой точностью

определять, насколько вредно некоторое исследуемое вещество. Судя по всему, показания датчиков слабо коррелируют с опасностью для живых существ, в частности, человека, и измеряют иные химико-физические характеристики.

5. ЗАКЛЮЧЕНИЕ

Выполнили: Анна Белова, Александр Лапшин

В рамках данного исследования группой познакомилась с системой «Электронный нос» и изучила поставляемые данные, был сделан вывод о том, что полученных исходных данных недостаточно для обучения, поэтому было предложено 2 подхода к генерации искусственных данных. Кроме того, обнаружено, что из всей матрицы 8 x 120 значимым является лишь один компонент, что позволяет свести каждый объект к 120-мерному вектору и сформировать датасет в виде матрицы, а не тензора. Затем был проведен сравнительный анализ классических алгоритмов машинного обучения и глубоких нейронных сетей и сделан вывод о том, что на полученной искусственной выборке с задачей n-label классификации хорошо справляется SVM, нейронные сети же показывают худший результат из-за переобучения. Таким образом, задачи 1 и 2 можно считать выполненными, а первый из ожидаемых результатов полностью получен и представлен в файле classification_report.xlsx, а задача 3 требует более детальной проработки для улучшения usability.

Для задачи визуализации разработана общая концепция на основе классов опасности и сделан вывод о границах применимости данной идеи.

Ввиду ограниченности временных ресурсов, не все поставленные задачи удалось завершить и реализовать в виде рабочих прототипов. Однако результаты, полученных в данном исследовании, открывают возможности для дальнейшей работы.