Министерство образования и науки Российской Федерации

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | | | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования | | | | | |
| «Алтайский государственный технический университет им. И.И. Ползунова» | | | | | |
| Факультет информационных технологий  Кафедра "Прикладная математика" | | | | | |
|  | | | Курсовой проект защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_\_ | | |
|  | | | Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С.М. Старолетов  Подпись | | |
|  | | | «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2017 г. | | |
|  | | |  | | |
| **ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОМУ ПРОЕКТУ** | | | | | |
| **Проектирование системы распознавания объектов различного типа с помощью каскада нейронных сетей** | | | | | |
| по дисциплине «Архитектурное проектирование и паттерны программирование» | | | | | |
| **КП 09.03.04.09.000 ПЗ** | | | | | |
| Студент группы ПИ-42 С.А. Киреков | | | | | |
|  |  |  | | (подпись) | и.о.,фамилия |
| Преподаватель доцент, к. ф-м н. С.М. Старолетов | | | | | |
| (должность, ученая степень) (подпись) | | | | | и.о.,фамилия |

Барнаул 2017

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Задание**  Учебная дисциплина: Архитектурное проектирование и паттерны программирование  ФИО студента: Киреков Семен Алексеевич Группа: ПИ-42  Тема курсового проекта: Проектирование системы распознавания объектов различного типа с помощью каскада нейронных сетей  Этапы разработки курсового проекта и сроки их выполнения:  1. Изучение необходимой учебной и научно-технической литературы (01.10.2017 – 30.10.2017);  2. Разработка приложения (30.10.2017 – 20.11.2017);   1. Оформление отчета о проделанной работе (20.11.2017 – 20.12.2017); 2. Сдача работы руководителю и защита работы (20.12.2017 – 28.12.2017).   Дата выдачи задания: 01.10.2017 Срок защиты: 28.12.2017  Руководитель: доцент С.М. Старолетов | | | | | | | | | | |
|  |  |  |  |  | КП 09.03.04.09.000 ПЗ | | | | | |
|  |  |  |  |  |
| Изм. | Лист | № докум. | Подп. | Дата |
| Разраб. | | Киреков С.А. |  |  | Проектирование системы распознавания объектов различного типа с помощью каскада нейронных сетей | Лит. | | | Лист | Листов |
| Пров. | | Старолетов С.М. |  |  |  | У |  | 2 | 29 |
|  | |  |  |  | АлтГТУ, ФИТ  *гр. ПИ-42* | | | | |
| Н.контр. | | Старолетов С.М. |  |  |
| Утв. | | Кантор С.А |  |  |

**Содержание**

[Введение 4](#_Toc501913404)

[1 Анализ проблемы и существующих методов решения 5](#_Toc501913405)

[1.1 Обзор предметной области 5](#_Toc501913406)

[1.2 Анализ существующих решений 10](#_Toc501913407)

[2 Описание программного обеспечения 12](#_Toc501913408)

[2.1 Структура реализованного приложения 12](#_Toc501913409)

[2.2 Применяемые паттерны 13](#_Toc501913410)

[2.3 Диаграмма классов 14](#_Toc501913411)

[2.4 Схема базы данных 14](#_Toc501913411)

[2.4 Дальнейшее развитие системы 17](#_Toc501913412)

[3 Результаты вычислительного эксперимента 18](#_Toc501913413)

4 Тезисы на научную конференцию………………………………………………...19

[Заключение 19](#_Toc501913414)

[Список использованных источников 28](#_Toc501913415)

[Приложение А. Код программы 29](#_Toc501913416)

# Введение

В настоящее время машинное обучение и, в частности, нейронные сети, являются одной из самых востребованных и быстроразвивающихся областей. Нейронные сети, а точнее искусственные нейронные сети, применяются в распознавании речи и преобразовании ее в текст, либо в набор команд, понятных системе (например, голосовое управление на современных смартфонах), аппроксимации различных явлений (предсказывания курсов валют), в медицинской диагностике (например, определение вероятности развития у человека сердечно-сосудистых заболеваний в ближайшее время на основании таких данных как средняя частота сердцебиения, скорость крови в сосудах, давлении и других) и во многих других областях.

# Анализ проблемы и существующих методов решения

## Обзор предметной области

Искусственная нейронная сеть (ИНС) — математическая модель, а также её программное или аппаратное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей — сетей нервных клеток живого организма.

ИНС представляет собой систему соединённых и взаимодействующих между собой простых процессоров (искусственных нейронов). Такие процессоры обычно довольно просты (особенно в сравнении с процессорами, используемыми в персональных компьютерах). Каждый процессор подобной сети имеет дело только с сигналами, которые он периодически получает, и сигналами, которые он периодически посылает другим процессорам. И, тем не менее, будучи соединёнными в достаточно большую сеть с управляемым взаимодействием, такие по отдельности простые процессоры вместе способны выполнять довольно сложные задачи.

ИНС применяют в тех задачах, когда универсальный алгоритм для их решения либо не существует, либо вывести его практически невозможно. Например, нам нужна система, которая на основе комментариев пользователя в социальных сетях, выдавала бы ему рекламу тех товаров, которые его могут заинтересовать. Составить детерменированный алгоритм для решения такой задачи не представляется возможным, так как комментарии могут быть вызваны огромных количеством разных факторов: настроение пользователя, возраст, пол, уровень образования, профессия и многих других. Уж не говоря о том, что в комментариях могут присутствовать ошибки (пунктационные, речевые, грамматические) или что сообщение может быть сарказмом и его нельзя понимать напрямую. Именно в ситуациях, когда алгоритм решения либо не существует, либо трудно выводим, применяются нейронные сети.

Стоит понимать, что нейронные сети работают с вероятностями, не являются детерменированными системами. То есть, нет гарантии, что, например, в ста процентах случаев пользователю будет подобрана корректная реклама. Однако в современных ИНС вероятность очень большая (в некоторых задачах может достигать 98-99 процентов), поэтому в данном случае их использование является оптимальным решением.

Нейронная сеть по сути является совокупностью многих нейронов, где выходные сигналы одних служат входными для других. Рассмотрим частной случай нейронной сети – один нейрон.

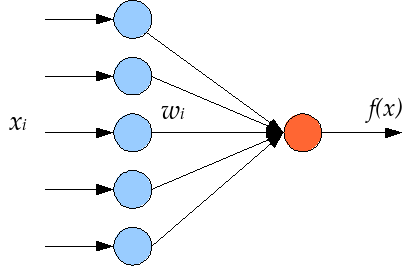


Рисунок 1.1: Схема нейрона

Нейрон действует по следующему принципу: на вход подаются сигналы Каждый из них умножается на *вес* То есть (стоит отметить, что как сигналы, так и веса могут быть нулевыми). Затем к полученному массиву числовых значений прибавляется смещение β и применяется функция которая называется *активационной*. Полученное значение и есть результат работы нейрона. Можно заметить, что выход у нейрона всегда один, в то время как входных значений может быть много. Входные принято называть *признаками*, а выходные – *ответами*. Очевидно, что на определенные входные данные мы рассчитываем получить определенный ответ. Для этого нужно правильно выбрать веса и смещение. Процесс подбора необходимых весов называется *обучением*. Различают *обучение с учителем* и *обучение без учителя*. Нейронные сети являются классическим примером обучения с учителем, так что рассмотрим этот вариант.

Для простоты понимания рассмотрим на конкретном примере. Пусть у нас есть *перцептрон* с двумя входами*.* Перцептрон – это нейрон, у которого активационная функция является *пороговой*. Общее описание пороговой функции: , где α – константа, а β – смещение. Для этой задачи положим α = 0. То есть, если сумма признаков, умноженных на соответствующие веса и плюс смещение больше 0, ответ 1, иначе 0. Пусть у нас есть следующий набор признаков и ответов:

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

По сути это является задачей нахождения конъюнкции между двумя числами. Одним из вариантов решения является следующий набор весов и смещения: Проверим:

Очевидно, что даже для такой простой задачи количество комбинаций правильных весов и смещения бесконечно. Например, мы можем умножить все веса и смещение на какую-либо константу (кроме нуля) и ничего не изменится.

Зачастую смещение также представляют в виде веса. Для этого нейрону добавляют еще один вход, на который всегда подается единица, а β убирают. С точки зрения математики ничего не меняется, однако это упрощает программирование.

Здесь веса были подобраны интуитивным образом, однако в прикладных задачах зачастую это не представляется возможным. Например, пусть у нас такая архитектура нейронной сети:

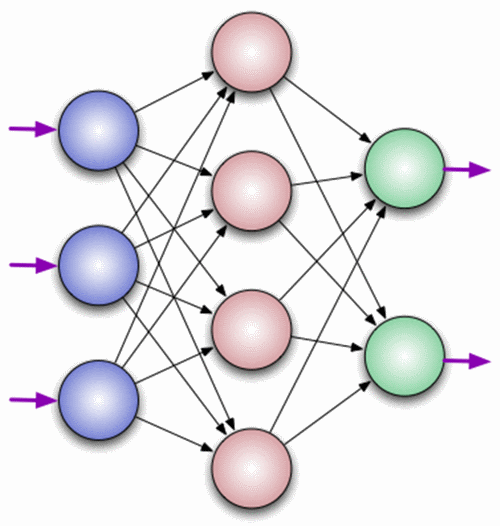


Рисунок 1.2: Концептуальная модель трехслойной нейронной сети.

Здесь представлена лишь концептуальная модель. Однако уже видно, что подобрать веса «на глаз» не получится. К тому же, для того, чтобы процесс обучения можно было запрограммировать, необходим детерменированный алгоритм. Для таких целей применяется *метод обратного распространения ошибки* и *метод градиентного спуска.*

## Анализ существующих решений

Так как в настоящее время нейронные сети приобрели большую популярность в решении различных задач, логично предположить, что есть библиотеки, которые упрощают процесс кодирования и не вынуждают писать все «с нуля». И это действительно так. Библиотек для машинного обучения и нейронных сетей огромное множество. Рассмотрим некоторые из них.

**Scikit-learn.**

Библиотека для машинного обучения для языка Python. Она не включает в себя нейронные сети, но, тем не менее, заслуживает внимания, так как в ней содержится множество других алгоритмов: метод ближайшего соседа, лассо, метод опорных векторов, метод k-средних, линейной регрессии и многих других.

**Tensorflow**

Библиотека с открытым исходным кодом от компании Google. Написана на языке C++. Есть реализации (API) для Python, Java, C++ и Go. Модели в tensorflow строятся на основе большого вычислительного графа, где вершины графа представляют собой математические операции, а дуги – многомерные массивы данных (тензоры). Граф строится с помощью API, а затем вычисляется разом с помощью обращения к функциям, написанным на C++. Это является принципиальным отличием от других библиотек, где обращение к «бэкэнду» производится каждый раз, когда необходимо произвести какое-то вычисление (например, библиотека numpy для python). Построение вычислительного графа позволяет снизить количество обращений до минимума и тем самым сильно повысить производительность. Tensorflow также позволяет распараллеливать вычисления между разными ядрами процессора или же выполнять вычисления на видеокарте (с помощью драйвера CUDA для Nvidia).

**Keras**

Keras не является самостоятельной библиотекой для машинного обучения. Он представляет из себя высокоуровневую обертку для языка Python, которая позволяет упростить работу с нейронными сетями и снизить количество кода. В качестве «бэкэнда» Keras может использовать Tensorflow, Theano или CNTK. Благодаря Keras даже сложные многоуровневые нейронные сети сможет написать человек без большой математической подготовки и глубокого понимания принципов работы ИНС.

# Описание программного обеспечения

## Структура реализованного приложения

Проектирование системы включает в себя несколько этапов:

Первый этап направлен на изучение предметной области и библиотек, которые могут быть использованы, а также выбор языка программирования. Исходя из этого, в качестве языка был выбран Python, а в качестве библиотек Tensorflow и Keras, также Orator – библиотека ORM для Python.

Второй этап включает в себя проектирование системы и ее разбиение на классы и интерфейсы. Проектирование выполнялось с учетом паттернов для возможного будущего расширения. Также сюда входить проектирование схемы базы данных, в которой будут хранится уже обученные нейронные сети для возможности дальнейшего обращения к ним. В качестве СУБД была выбрана MySQL, так как она является бесплатной и не требует большого количества ресурсов для локального запуска.

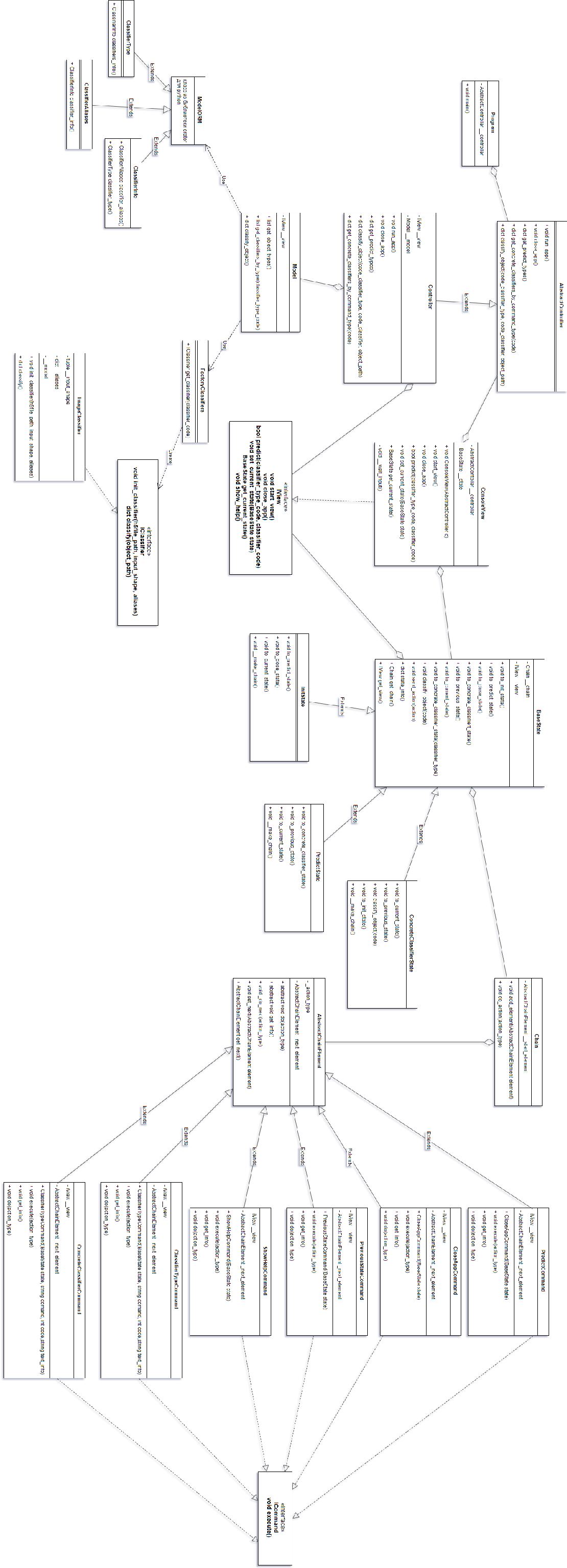
Третий этап включает в себя кодирование и тестирование.

## Применяемые паттерны

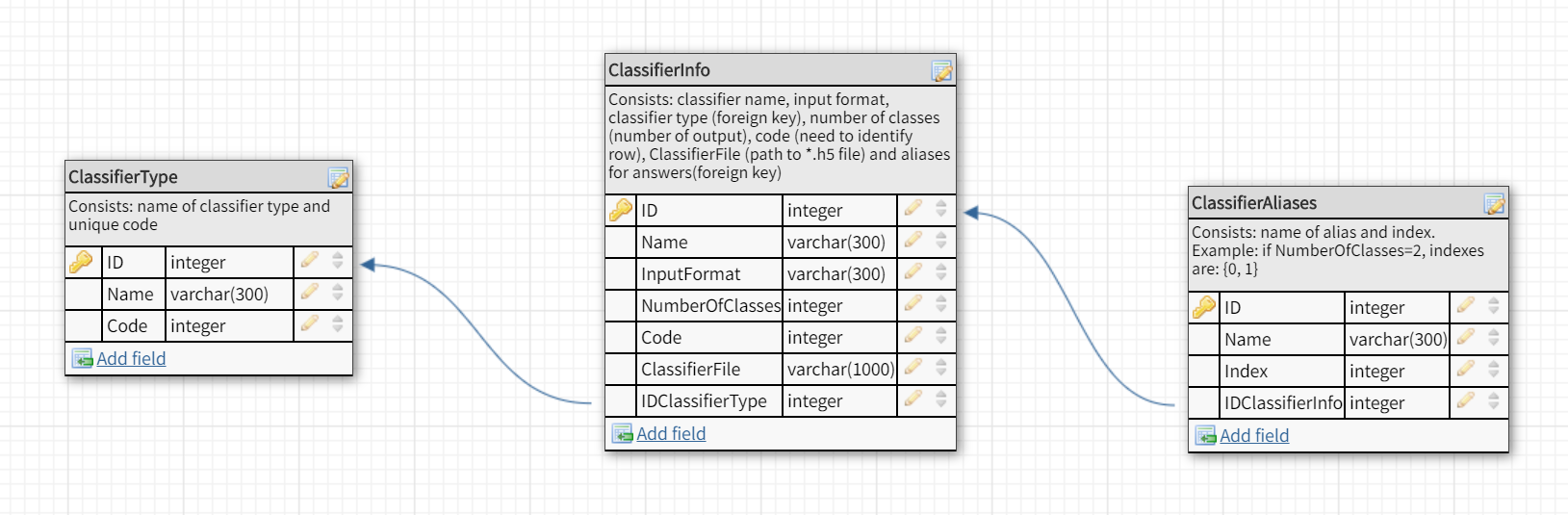
При написании приложения было использовано пять паттернов:

* MVC, который является фундаментальным для многих приложений (особенно часто применяется в веб-программировании). За счет разделения программы на три независимых части: модель (оперирует с данными, с базой данных), представление (GUI или же интерфейс командной строки, с которым взаимодействует пользователь) и контроллер (является связующим звеном между моделью и представлением), мы получаем возможность разрабатывать каждую часть по отдельности. К тому же, за счет низкой связности, изменение, например, представления, не приводит к изменению модели.
* Команда – паттерн, который позволяет инкапсулировать запрос на выполнение определенного действия в виде отдельного объекта. В качестве команды выступает выбор пункта меню.
* Цепочка Обязанностей позволяет избежать жесткой привязки отправителя запроса к получателю, позволяя нескольким объектам обработать запрос. В данном случае цепочка обязанностей хранит команды.
* Состояние позволяет объекту изменять свое поведение в зависимости от внутреннего состояния. Так как меню в программе многоуровневое, то каждое состояние есть текущая ветка и уровень вложенности меню. Каждое состояние имеют свою цепочку обязанностей команд.
* Фабрика - это способ создания объекта одного из нескольких возможных классов, основываясь на представленных данных. Подразумевается, что программа классифицирует объекты различных типов. Соответственно, в зависимости от того типа, который был выбран пользователь, возвращается соответствующий объект.

## Диаграмма классов



## 2.4.Схема базы данных



16

## Дальнейшее развитие системы

В дальнейшем можно ввести новые типы распознаваемых объектов, добавить новые классификаторы. Также добавить новые возможности в интерфейс программы, например, динамическое добавление новых классификаторов, возможность формирования отчетов по входным данным в различных форматах. Можно будет заменить интерфейс командной строки на GUI.

# Результаты вычислительного эксперимента

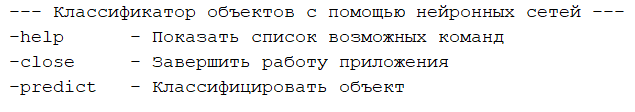


Рисунок 3.1: Начальное меню

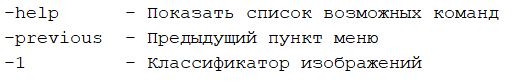


Рисунок 3.2: Меню выбора типа классификатора

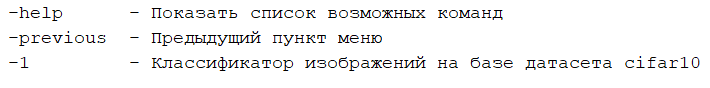


Рисунок 3.3: Меню выбора конкретного классификатора

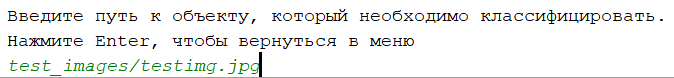


Рисунок 3.4: Выбор классифицируемого объекта

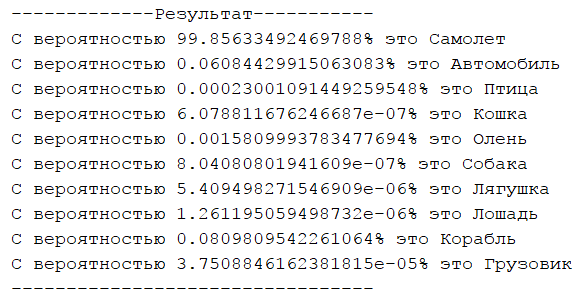


Рисунок 3.5: Результат

# Тезисы на научную конференцию

# Классификация различных объектов с помощью искусственных нейронных сетей.

# В настоящее время машинное обучение и, в частности, нейронные сети, являются одной из самых востребованных и быстроразвивающихся областей. Нейронные сети применяются в распознавании речи (например, голосовое управление на современных смартфонах), аппроксимации различных явлений (предсказывании курсов валют), в медицинской диагностике (например, в определении вероятности развития у человека сердечно-сосудистых заболеваний в ближайшее время на основании таких данных как средняя частота сердцебиения, скорость крови в сосудах, давлении и других) и во многих других областях.

# Искусственная нейронная сеть (ИНС) — математическая модель, а также её программное или аппаратное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей — сетей нервных клеток живого организма.

# ИНС представляет собой систему соединённых и взаимодействующих между собой простых процессоров (искусственных нейронов). Такие процессоры обычно довольно просты (особенно в сравнении с процессорами, используемыми в персональных компьютерах). Каждый процессор подобной сети имеет дело только с сигналами, которые он периодически получает, и сигналами, которые он периодически посылает другим процессорам. И, тем не менее, будучи соединёнными в достаточно большую сеть с управляемым взаимодействием, такие по отдельности простые процессоры вместе способны выполнять довольно сложные задачи.

# ИНС применяют в тех задачах, в которых универсальный алгоритм для их решения либо не существует, либо вывести его практически невозможно. Например, нам нужна система, которая на основе комментариев пользователя, выдавала бы ему рекламу тех товаров, которые его могут заинтересовать. Составить детерменированный алгоритм для решения такой задачи очень трудно, так как комментарии могут быть вызваны огромным количеством разных факторов: настроение пользователя, возраст, пол, уровень образования, профессия и многих других. Уж не говоря о том, что в комментариях могут присутствовать ошибки (пунктационные, речевые, грамматические) или что сообщение может быть сарказмом и его нельзя понимать напрямую. Именно в ситуациях, когда алгоритм решения либо не существует, либо трудно выводим, применяются нейронные сети.

# Нейронная сеть, по сути, является совокупностью многих нейронов, где выходные сигналы одних служат входными для других. Рассмотрим частной случай нейронной сети – один нейрон.

# https://upload.wikimedia.org/wikipedia/ru/b/ba/Single_layer_perceptron.png

# Рисунок 1 – Схема одного искусственного нейрона

# Нейрон действует по следующему принципу: на вход подаются сигналы . Каждый из них умножается на *вес* . То есть (стоит отметить, что как сигналы, так и веса могут быть нулевыми). Затем полученный массив числовых значений суммируется, к нему прибавляется *смещение* β и к полученному общему значению применяется функция f(x), которая называется *активационной*. Результат функции и есть результат работы нейрона. Можно заметить, что выход у нейрона всегда один, в то время как входных значений может быть много. Входные принято называть *признаками*, а выходные – *ответами*. Очевидно, что на определенные входные данные мы рассчитываем получить определенный ответ. Для этого нужно правильно выбрать веса и смещение. Процесс подбора необходимых весов называется *обучением*. Различают *обучение с учителем* и *обучение без учителя*. Рассмотрим первый вариант, так как именно он является предметом моего исследования.

Для простоты понимания рассмотрим на конкретном примере. Пусть у нас есть *перцептрон* с двумя входами*.* Перцептрон – это нейрон, у которого активационная функция является *пороговой*. Общее описание пороговой функции: , где α – константа, а β – смещение. Для этой задачи положим α = 0. То есть, если сумма признаков, умноженных на соответствующие веса и плюс смещение больше либо равно 0, ответ 1, иначе 0. Пусть у нас есть следующий набор признаков и ответов:

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

По сути это является задачей нахождения конъюнкции между двумя числами. Одним из вариантов решения является следующий набор весов и смещения: Проверим:

Очевидно, что даже для такой простой задачи количество вариантов правильных весов и смещения бесконечно. Например, мы можем умножить все веса и смещение на какую-либо константу (кроме нуля) и ничего не изменится.

Зачастую смещение также представляют в виде веса. Для этого нейрону добавляют еще один вход, на который всегда подается единица, а β убирают. С точки зрения математики ничего не меняется, однако это упрощает программирование.

Предметом моего исследования является проектирование и разработка десктопного приложения, которое позволит:

• Классифицировать различные объекты с помощью нейронных сетей

• На каждый тип классифицируемых объектов может приходиться несколько различно обученных классификаторов

• Информация о классификаторах хранится в базе данных

• Приложение должно предоставлять понятный и удобный пользователю GUI

• Приложение должно быть разработано с учетом паттернов проектирования, чтобы оставалась возможность его дальнейшего расширения и замены отдельных частей

• Приложение должно быть кроссплатформенным

Первым делом необходимо ознакомиться с библиотеками для работы с нейронными сетями, которые на текущий момент доступны и поддерживаются. Изучив данный вопрос, выбор пал на язык программирования Python следующие библиотеки:

**Tensorflow**

Библиотека с открытым исходным кодом от компании Google. Написана на языке C++. Есть реализации (API) для Python, Java, C++ и Go. Модели в tensorflow строятся на основе большого вычислительного графа, где вершины графа представляют собой математические операции, а дуги – многомерные массивы данных (тензоры). Граф строится с использованием API, а затем вычисляется разом с помощью обращения к функциям, написанным на C++. Это является принципиальным отличием от других библиотек, где обращение к «бэкэнду» производится каждый раз, когда необходимо произвести какое-то вычисление (например, библиотека numpy для python). Построение вычислительного графа позволяет снизить количество обращений до минимума и тем самым сильно повысить производительность. Tensorflow также позволяет распараллеливать вычисления между разными ядрами процессора или же выполнять вычисления на видеокарте (с помощью драйвера CUDA для Nvidia).

**Keras**

Keras не является самостоятельной библиотекой для машинного обучения. Он представляет из себя высокоуровневую обертку для языка Python, которая позволяет упростить работу с нейронными сетями и снизить количество кода. В качестве «бэкэнда» Keras может использовать Tensorflow, Theano или CNTK. Благодаря Keras даже сложные многоуровневые нейронные сети сможет написать человек без большой математической подготовки и глубокого понимания принципов их работы.

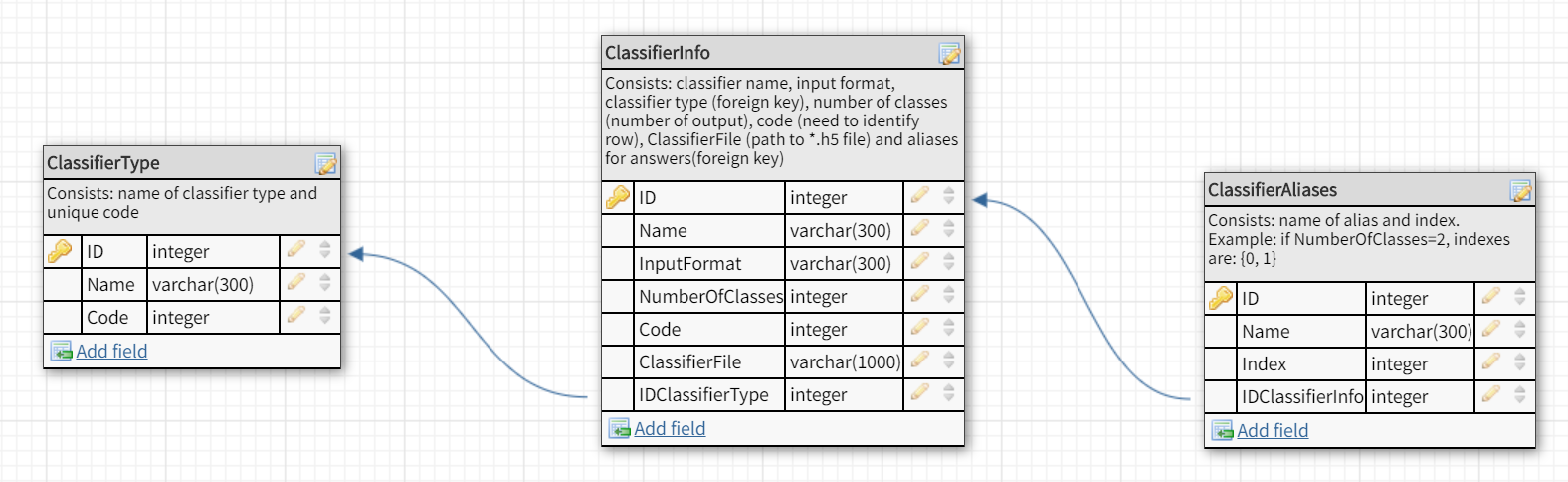
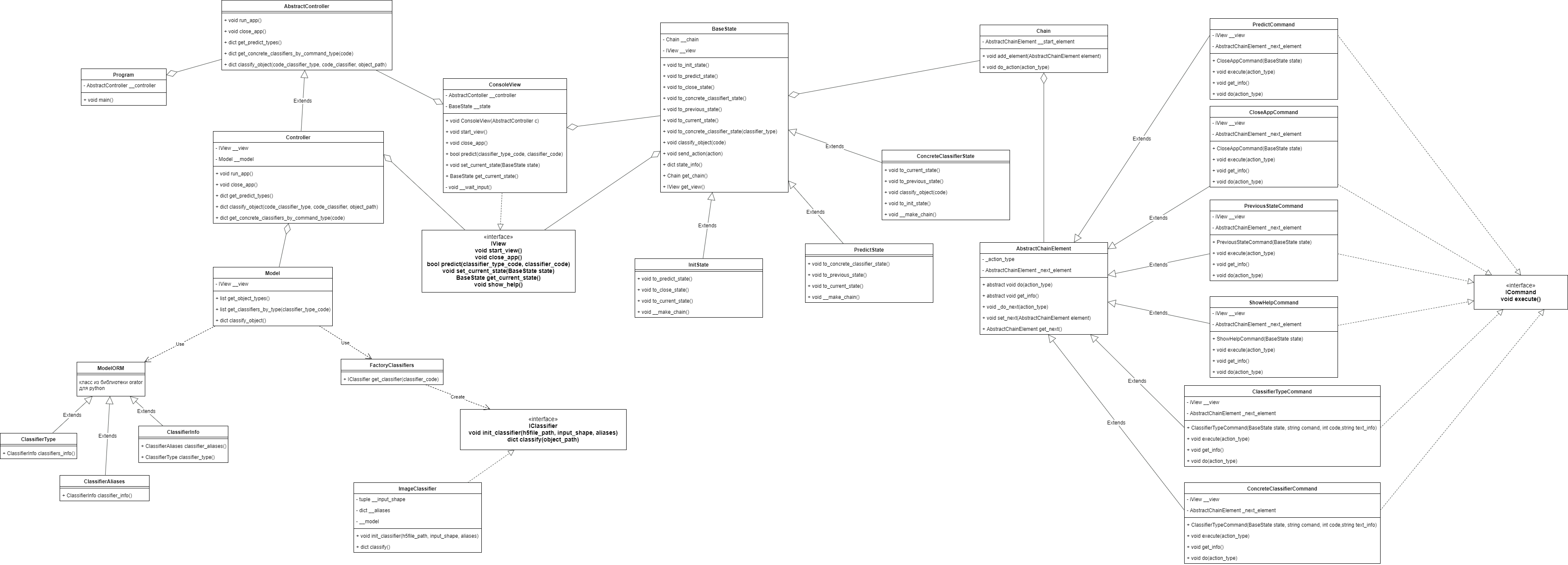
Следующим важным моментом является проектирование схемы базы данных, то есть определение того, как именно обученные нейросети будут хранится. В данный момент схема выглядит таким образом:

Рисунок 2 – Схема базы данных.

То есть мы храним тип классификатора в таблице ClassifierType (например, классификация изображений или классификация музыки по жанрам), у которого может быть много конкретных классификаторов (обученных нейросетей). В таблице ClassifierInfo (конкретный классификатор) мы храним относительный путь к файлу обученной нейросети. Интерфейс библиотеки Keras позволяет сохранить всю архитектуру нейросети вместе с весами в один файл с расширение \*.h5, что позволяет хранить в базе данных любые обученные нейронные сети. Таблица ClassifierAliases хранит алиасы (псевдонимы) для названий объектов, которые может классифицировать данная нейронная сеть. Теперь рассмотрим текущую архитектуру приложения:



# Рисунок 3 – Диаграмма классов приложения

# При написании приложения было использовано пять паттернов:

# • *MVC*, который является фундаментальным для многих приложений (особенно часто применяется в веб-программировании). За счет разделения программы на три независимых части: модель (оперирует с данными, с базой данных), представление (GUI или же интерфейс командной строки, с которым взаимодействует пользователь) и контроллер (является связующим звеном между моделью и представлением), мы получаем возможность разрабатывать каждую часть по отдельности. К тому же, за счет низкой связности, изменение, например, представления, не приводит к изменению модели. В данном случае модель обращается к базе данных и по запросу контроллера возвращает необходимую информацию.

# • *Команда* – паттерн, который позволяет инкапсулировать запрос на выполнение определенного действия в виде отдельного объекта. В качестве команды выступает выбор пункта меню.

# • *Цепочка обязанностей* позволяет избежать жесткой привязки отправителя запроса к получателю, позволяя нескольким объектам обработать запрос. В данном случае цепочка обязанностей хранит команды.

# • *Состояние* позволяет объекту изменять свое поведение в зависимости от внутреннего состояния. Так как меню в программе многоуровневое, то каждое состояние есть текущая ветка и уровень вложенности меню. Каждое состояние имеют свою цепочку обязанностей команд.

# • *Фабрика* - это способ создания объекта одного из нескольких возможных классов, основываясь на представленных данных. Подразумевается, что программа классифицирует объекты различных типов. Соответственно, в зависимости от того типа, который был выбран пользователь, возвращается соответствующий объект.

# На текущий момент приложение имеет консольный интерфейс, однако, учитывая тот факт, что приложение было спроектировано с учетом паттернов, заменить его на GUI не составит труда. В будущем также планируется добавления следующих функциональных возможностей:

# Добавление новых типов классификаторов (на данный момент приложение умеет классифицировать только изображения)

# Возможность добавления новых классификаторов в интерактивном режиме (сейчас, чтобы добавить новый классификатор, приходится напрямую обращаться к базе данных)

# Добавление поддержки английского языка для интерфейса приложения (на текущий момент присутствует только русский)

# Добавление графического интерфейса, при этом оставив возможность запускать приложение в режиме консоли

# Список использованных источников

1. Машинное обучение и анализ данных [Электронный ресурс]. - Режим доступа: <https://ru.coursera.org/specializations/machine-learning-data-analysis>.
2. Нейронные сети [Электронный ресурс]. - Режим доступа: https://stepik.org/course/401/
3. Хайкин Саймон. Нейронные сети. Полный курс. — Москва: Издательство Вильямс, 2016.— С. 1104. — ISBN 978-5-8459-2069-0.

# Заключение

Таким образом, приложение, дает возможность классифицировать изображения, а также добавлять новые классификаторы.

В качестве дальнейшего расширения системы, можно добавить новые типы классифицируемых объектов и графический интерфейс.

# Список использованных источников

1. Машинное обучение и анализ данных [Электронный ресурс]. - Режим доступа: <https://ru.coursera.org/specializations/machine-learning-data-analysis>.
2. Нейронные сети [Электронный ресурс]. - Режим доступа: https://stepik.org/course/401/
3. Хайкин Саймон. Нейронные сети. Полный курс. — Москва: Издательство Вильямс, 2016.— С. 1104. — ISBN 978-5-8459-2069-0.

# Приложение А. Код программы

Код программы находится на приложенном диске.