МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

РАБОТА ПРОВЕРЕНА
Рецензент,
Директор
ООО "Napoleon IT"
_____ П.С.Подкорытов
“___”_____ 2019 г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой,
д.ф.-м.н., профессор
_____ Л.Б.Соколинский
“___”_____ 2019 г.

РАЗРАБОТКА ИГРОВОГО ПРИЛОЖЕНИЯ ДЛЯ ОБУЧЕНИЯ ОСНОВАМ РАБОТЫ НЕЙРОННЫХ СЕТЕЙ

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 02.03.02.2019.115-106.ВКР

Научный руководитель:
кандидат физ.-мат. наук, доцент
_____ Г.И. Радченко

Автор работы,
студент группы КЭ-401

_____ Л.А. Боровинская

Ученый секретарь

(нормоконтролер)

_____ О.Н. Иванова

“___”_____ 2019г.

Челябинск-2019

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
«___»._____.2019

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра

студентке группы КЭ-401
Боровинской Любови Александровне,
обучающейся по направлению
02.03.02 «Фундаментальная информатика и информационные технологии»

1. **Тема работы** (утверждена приказом ректора от 25.04.2019 № 899)
   Разработка игрового приложения для обучения основам работы нейронных сетей
2. **Срок сдачи студентом законченной работы:** 27.05.2019.
3. **Исходные данные к работе**

3.1. Рашид, Т. Создай свою нейросеть – М.: Вильямс, 2017. – 272 с.
3.2. А. Торн Основы анимации в Unity – Изд-во ДМК, 2016. – 176 с.
3.3. Михайленко Т. М. Игровые технологии как вид педагогических технологий //Материалы Междунар. науч. конф. (г. Челябинск, октябрь 2011 г.). Т. I. – Челябинск: Два комсомольца, 2011. – с. 140-146.

4. **Перечень подлежащих разработке вопросов**

4.1. Произвести подбор литературы, необходимой для разработки программы.
4.2. Выполнить анализ существующих решений.
4.3. Определить ключевые требования программы.
4.4. Спроектировать, реализовать и протестировать программу.

5. **Дата выдачи задания:** 26.11.2019.

Научный руководитель
Кандидат физ.-мат. наук, доцент Г.И. Радченко
Задание принял к исполнению Л.А. Боровинская

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Современный мир – это мир компьютерных технологий. Сейчас уже сложно представить жизнь без них. Развитие компьютерных технологий привело к тому, что во многом компьютеры стали заменять людей.

В двадцать первом веке люди настолько развили науку и технологии, что современная техника может выполнять такие задачи, решение которых заняло бы у человека годы и потребовало больших затрат как денежных средств, так и трудовых ресурсов.

Однако существует ряд задач, легких с точки зрения человека, но сложных для компьютера. К таким задачам относят классификацию изображений, например, распознавание рукописных символов.

Чтобы научить компьютер решать подобные задачи используется машинное обучение – класс методов искусственного интеллекта, характерной чертой которых является не прямое решение задачи, а обучение в процессе применения решений множества сходных задач. Однако, разработка искусственного интеллекта достаточно сложна в освоении.

Актуальность темы исследования

На сегодняшний момент существует некоторое количество курсов и платформ, обучающих работе с нейронными сетями, на основе которых и создается искусственный интеллект, но практически у всех из них очень высокий порог вхождения: требуются знания высшей математики, библиотек машинного обучения. Кроме того, основная масса ресурсов – это статьи и книги, и практически все они на английском языке, что является сложным препятствием для многих желающих изучить нейронные сети. Обучающих интерактивных приложений, которые могли бы на практике познакомить с работой нейронных сетей, фактически нет.

Исходя из вышесказанного, на сегодняшний день разработка игрового приложения, обучающего основам работы нейронных сетей, является актуальной задачей.

Цель и задачи проекта

В ходе проекта необходимо разработать игровое приложение для обучения основам работы нейронных сетей. Для выполнения поставленной цели необходимо решить следующие задачи:

1) произвести подбор литературы, необходимой для разработки программы;
2) выполнить анализ существующих решений;
3) определить ключевые требования программы;
4) спроектировать, реализовать и протестировать программу.

Структура и объем работы

Работа состоит из введения, 4 разделов, заключения, списка использованной литературы, приложения.

Работа составляет 37 страниц, в списке литературы указано 20 источников, объем приложения – 5 страниц.

В первой главе производится обзор научных работ по тематике исследования и анализ предметной области, приведен обзор существующих аналогов.

Во второй главе раскрывается проектирование приложения, его компонентов, приведены поясняющие диаграммы.

Третья глава описывает разработку программ-компонентов приложения, объединение их в единую систему.

В четвертой главе приведены таблицы с результатами функционального и интеграционного тестирования компонентов.

В заключении выведены основные результаты проделанной работы.

В приложении располагаются листинги некоторых участков кода приложения.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Обучение основам работы нейронных сетей

Разработка искусственного интеллекта достаточно сложна в освоении. На сегодняшний момент существует некоторое количество курсов и платформ, обучающих работе с нейронными сетями, на основе которых и создается искусственный интеллект, но у одних очень высокий порог вхождения: требуются знания высшей математики, библиотек машинного обучения, другие раскрывают далеко не все технологии создания искусственного интеллекта. Кроме того, большинство ресурсов – это статьи и книги, обучающих интерактивных приложений практически нет или они на английском языке, что является сложным препятствием для многих желающих изучить нейронные сети.

Контсаренко Ф. делится своим опытом создания интерфейса обучающего приложения [8]. Автор рассказывает о том, как с помощью интерфейса сделать процесс обучения более занимательным. В статье так же приводятся примеры методов интерактивного обучения. Из статьи можно вынести основные требования, на которых будет базироваться дальнейшая разработка обучающего приложения.

Из выступления Михайленко Т.М. на международной научной конференции с докладом «Игровые технологии как вид педагогических технологий» [10] можно вывести необходимые условия использования игровых технологий.

В статьях «Игры, которые учат программированию» [5], «10 игр для изучения программирования» [20] и «Необычные игры, которые учат программированию и логике» [7] приведены примеры и описания игровых приложений, обучающих программированию. Данные материалы раскрывают особенности обучающих игр, их положительные стороны и недостатки.

В статье «Как добиться того, чтобы обучение в играх не раздражало» [6] подробно описываются примеры обучения из разных игр. Примеры

снабжены комментариями и изображениями, содержат детально описанные положительные и отрицательные моменты, которые соответственно упрощают или усложняют процесс обучения.

Приведенные выше статьи и книги помогают сформировать список требований и основные характеристики обучающего приложения:

1) приложение должно быть ориентировано на определенную аудиторию и учитывать уровень сложности и трудоемкости для людей с определенным уровнем подготовки;

2) дизайн системы обучения следует делать легким, запоминающимся и располагающим к дальнейшему получению знаний;

3) цвета не должны быть отторгающими или вызывающими чувство раздражения и неприятия;

4) интерфейс должен быть ненавязчивым и легким в освоении;

5) в приложении должны быть задействованы технологии интерактивного обучения, использованы примеры и описания;

6) информация не должна быть тяжелой для восприятия, содержать большие объемы.

1.2. Обзор аналогов

Рассмотрим ресурсы, обучающие основам нейронных сетей. В книге Рашида Т. «Создай свою нейросеть» [19] и в материале «Распознавание оптических образов (символов) с помощью однослойного персептрона» [18] очень хорошо раскрыто устройство нейронных сетей, описаны принципы их работы, того, как сети обучаются. Однако данные ресурсы предоставляют прежде всего теоретические знания.

В обзорах курсов по машинному обучению [12] и нейронным сетям [4] описываются платформы, курсы и приложения, с помощью которых можно изучить основы построения нейронных сетей, приводятся достоинства и недостатки ресурсов. Рассмотрим некоторые из них.

TensorFlow Playground [3] – веб-приложение, визуализирующее процесс обучения нейронной сети, классифицирующей точки на координатной плоскости (рис. 1). Точки данных (представленные маленькими кружками) изначально окрашены в оранжевый или синий цвета, которые соответствуют положительному и отрицательному. В выходном слое точки окрашиваются в оранжевый или синий цвета в зависимости от их исходных значений. Цвет фона показывает, что сеть предсказывает для конкретной области. Интенсивность цвета показывает, насколько достоверен этот прогноз. Данное приложение демонстрирует работу НС, но не поясняет ее принципы работы, не раскрывает, как устроена сеть с точки зрения кода, кроме того, очень мало поясняющего материала.

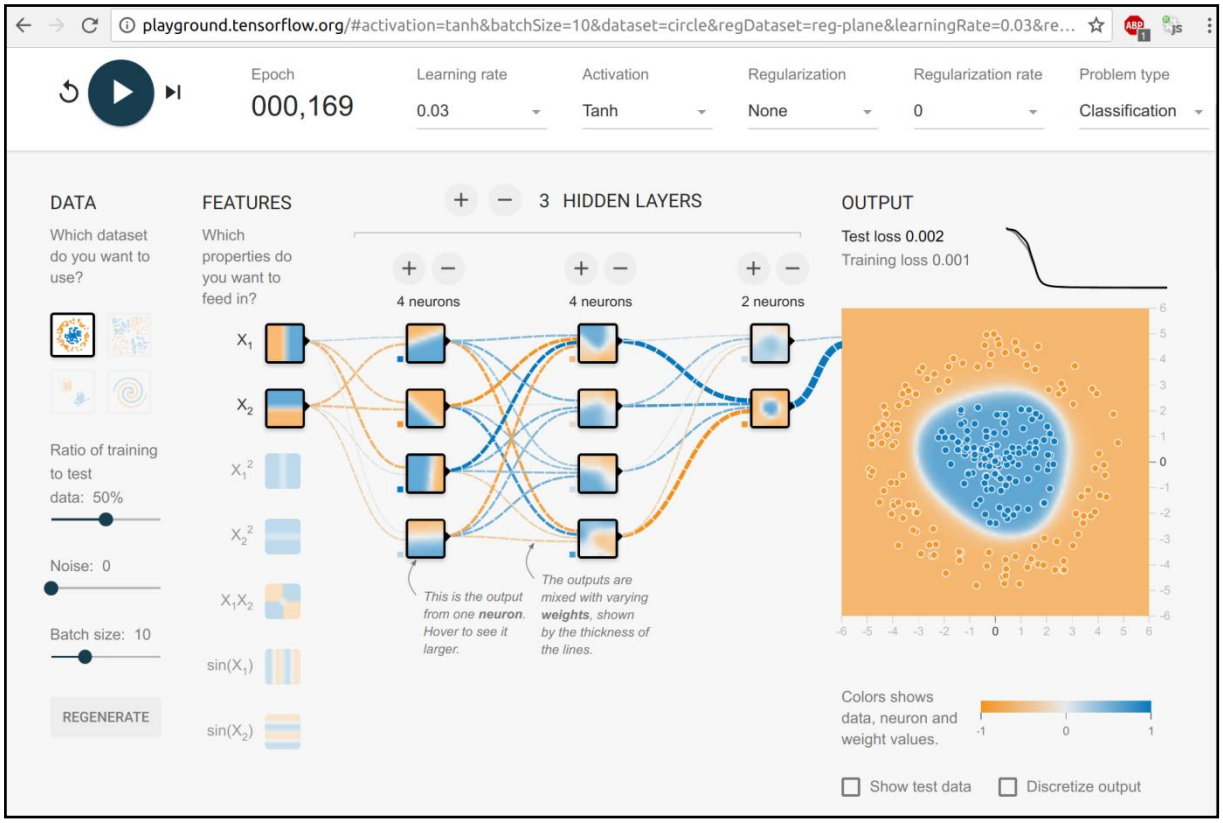


Рис. 1. Интерфейс TensorFlow Playground

Алгоритм машинного обучения Flappy Bird [1], использующий нейронные сети и генетический алгоритм. Это небольшая игра, написанная на HTML5. Есть описание используемой нейронной сети, генетического алгоритма, приведена ссылка на исходный код. Интерфейс приведен на рис. 2.

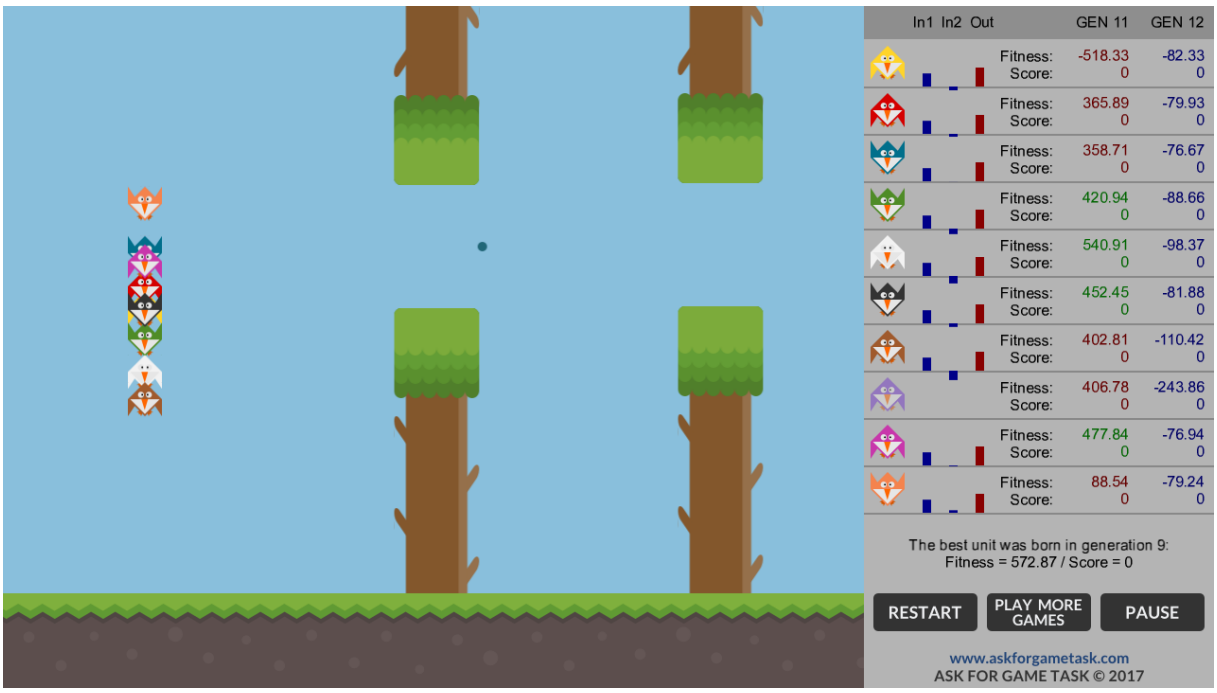


Рис. 2. Интерфейс приложения Flappy Bird

Курс «Deep Learning» от Google[9] (рис. 3).

Преимущества:

- большое количество интерактивных опросов и практических заданий в формате iPython Notebook + курсовой проект: разработка мобильного приложения, распознающего цифры на изображениях с камеры телефона в реальном времени;
- возможность обсуждения заданий с преподавателями и другими студентами;
- программа не перегружена, покрывает как общую теорию нейронных сетей, так и их основные типы, используемые в глубоком обучении.

Недостатки:

- курс на английском языке;
- достаточно высокая сложность: курс имеет ограниченную вводную часть, которая охватывает только часть базовых тем машинного обучения;

- в бесплатной версии все практические задания проверяются самим учащимся на основе результатов работы моделей и метрик качества, получение комментариев от преподавателей или других студентов не предусмотрено.

Рис. 3. Презентация курса «Deep Learning»

1.3. Общие сведения о нейронных сетях

Сверточные НС – это сети с особой архитектурой, разработанной для задач распознавания и классификации изображений. Согласно статье Москалева [11], данная архитектура является одной из наиболее эффективных для решения задач классификации изображений и работы с графической и аудио информацией. Особенность данной архитектуры заключается в том, что она обрабатывает данные фрагментами, но информация не дробится на части, а происходит последовательный прогон.

Нейронная сеть – это последовательность нейронов, соединенных между собой синапсами-связями. Структура искусственных НС появилась из биологии. Благодаря такой структуре, машина получает возможность

анализировать и запоминать различную информацию. Другими словами, искусственная нейронная сеть – это программа, работающая по тем же принципам, что и мозг человека, в котором находятся множество нейронов передающих информацию в виде электрических импульсов.

Нейрон – это некоторая вычислительная единица, которая получает импульс – некоторую информацию, представленную в виде числа в диапазоне от нуля до единицы, проводит над ней определенные вычисления и передает ее по синапсам нейронам на следующем слое. Нейроны классифицируют на три вида: входной, скрытый и выходной.

Входной слой – это слой, который получает информацию, скрытые слои – слои, которые ее обрабатывают, и выходной слой – слой, который выводит результат.

У каждого из нейронов есть 2 основных параметра: входные данные и выходные данные. У входных нейронов входные и выходные данные совпадают. У нейронов скрытых слоев и выходного слоя в поле входных данных попадает суммарная информация всех нейронов с предыдущего слоя. Данная информация-сигнал корректируется с помощью некоторой функции и попадает в поле выходных данных.

Синапс – это связь между двумя нейронами. Каждый синапс имеет свой вес – некоторое число, благодаря которому входная информация изменяется, когда передается от нейронов одного слоя к нейронам другого слоя. Пример нейронной сети изображен на рис. 4.

В настоящее время нейронные сети используются для решения задач с высокой сложностью, требующих аналитических вычислений подобных тем, что делает человеческий мозг. Наиболее распространенными применениями НС являются следующие задачи.

Задачи распознавания – используются для поиска изображений и фотографий, для распознавания лиц, в системах безопасности.

Задачи классификации – применяются для распределения данных по категориям. Например, нейронная сеть, анализируя возраст, платежеспо-

собность и кредитную историю может определить, давать кредит человеку или нет.

Задачи предсказания – НС могут предсказывать следующий шаг. Например, рост или падение акций на основе ситуации на фондовом рынке, какие действия следует предпринять персонажу, управляемому сетью, чтобы никогда не проигрывать.

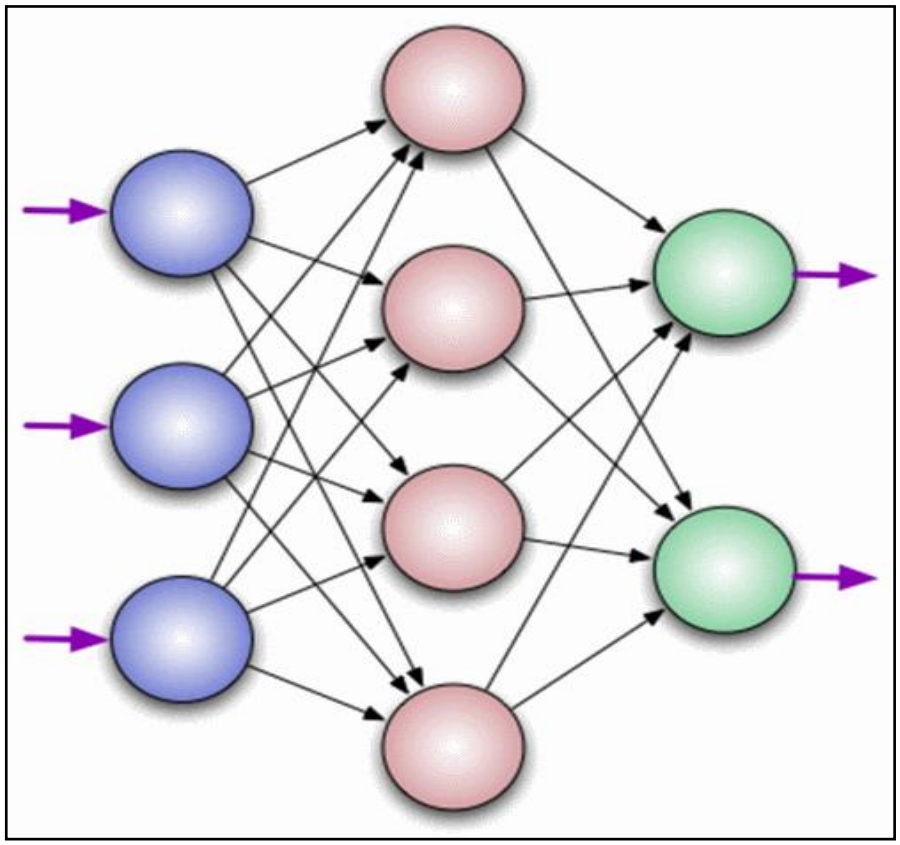


Рис. 4. Схема нейронной сети

Обучение нейронной сети, использованной в данном проекте, будет происходить по алгоритму обратного распространения ошибки. Данный алгоритм предполагает два прохода по всем слоям сети: прямого и обратного. При прямом проходе входной вектор подается на входной слой нейронной сети, после чего распространяется по сети от слоя к слою. В результате генерируется набор выходных сигналов, который и является фактической реакцией сети на данный входной образ. Во время прямого прохода все синаптические веса сети фиксированы. Во время обратного про-

хода все синаптические веса настраиваются в соответствии с правилом коррекции ошибок, а именно: фактический выход сети вычитается из желаемого, в результате чего формируется сигнал ошибки. Этот сигнал впоследствии распространяется по сети в направлении, обратном направлению синаптических связей.

2. ПРОЕКТИРОВАНИЕ

2.1. Общие сведения

Обозначение и наименование программы:

Интерактивное приложение, обучающее основам работы нейронных сетей.

Обучающее приложение состоит из двух частей: нейронная сеть на языке Python, распознающая рукописные числа, и игровое приложение, знакомящее пользователя с обучением сети на практике.

Программное обеспечение, необходимое для функционирования программы:

- браузер Google Chrome, Mozilla Firefox;
- платформы разработки – Unity, PyCharm.

Программа написана на языках C# иPython.

Нейронные сети обоих программ должны быть реализованы на основе алгоритма обратного распространения ошибки.

2.2. Определение требований к приложению

Для создаваемой системы будут установлены следующие функциональные требования:

1) система должна давать понимание устройства и работы нейронных сетей;

2) обучение должно быть построено в игровой форме для упрощения понимания теории;

3) система должна давать возможность пользователю работать непосредственно с самой нейронной сетью и кодом, реализующим ее.

Нефункциональные требования к создаваемому приложению:

1) система должна быть интерактивной;

2) система должна быть реализована в формате веб-приложения;

3) язык интерфейса должен быть русским;

4) система должна поддерживать различные браузеры, то есть должна быть кроссплатформенной.

Приложение должно использоваться для обучения и закрепления знаний по основам работы нейронных сетей, а также при проведении теоретических и практических автоматизированных учебных занятий.

2.3. Варианты использования игрового приложения для обучения основам работы нейронных сетей

Для описания вариантов использования и внешних требований к системе была взята диаграмма вариантов использования – диаграмма, отражающая отношения между актерами и прецедентами.

Разработанное приложение может использоваться для индивидуального обучения студентов и школьников основам работы нейронных сетей. На рис. 5 представлена диаграмма использования разрабатываемого приложения.

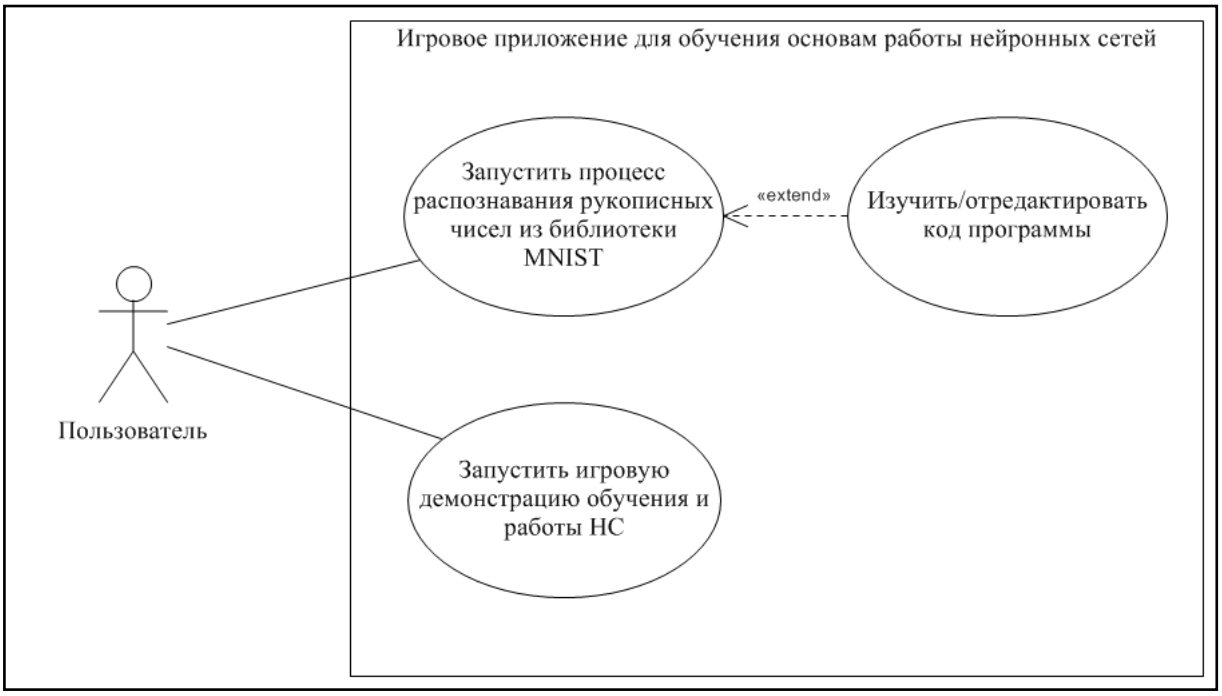


Рис. 5. Диаграмма использования разрабатываемого приложения

Пользователю приложения предоставлены следующие возможности:

1) запустить процесс распознавания рукописных чисел из библиотеки MNIST – пользователь может написать команду в консоли Jupyter Notebook, которая запустит обучение нейронной сети, затем, после того как сеть будет обучена, программа начнет выводить распознаваемые числа в формате: веса выходных нейронов, результат распознавания и вероятность верного результата, изображение числа, которое проходило распознавание;

2) изучить/отредактировать код программы – с помощью команд Jupyter Notebook пользователь вывести любой блок кода программы распознавания чисел, отредактировать его и запустить;

3) запустить игровую демонстрацию обучения и работы НС – помимо блоков кода программы, распознающей рукописные числа, в приложении будет выведено окно с демонстрационной игрой, благодаря которой пользователь сможет ознакомиться с работой нейронной сети в процессе ее обучения.

Данное приложение может использоваться для обучения и закрепления знаний по основам работы нейронных сетей, а также при проведении теоретических и практических автоматизированных учебных занятий**.**

2.4. Компоненты системы

В игровом приложении для обучения основам работы нейронных сетей имеются следующие компоненты:

1) программа распознавания рукописных чисел;

2) приложение, демонстрирующее обучение и работу нейронной сети в режиме реального времени.

Оба компонента объединены посредством интерфейса обучающего приложения (рис. 6). Данные компоненты не взаимодействуют между собой, но представляют единую систему обучающего приложения.

Пользователь может взаимодействовать с обоими компонентами через интерфейс обучающего приложения. Ни один из компонентов, исключая интерфейс, не могут повлиять на работу другого компонента.

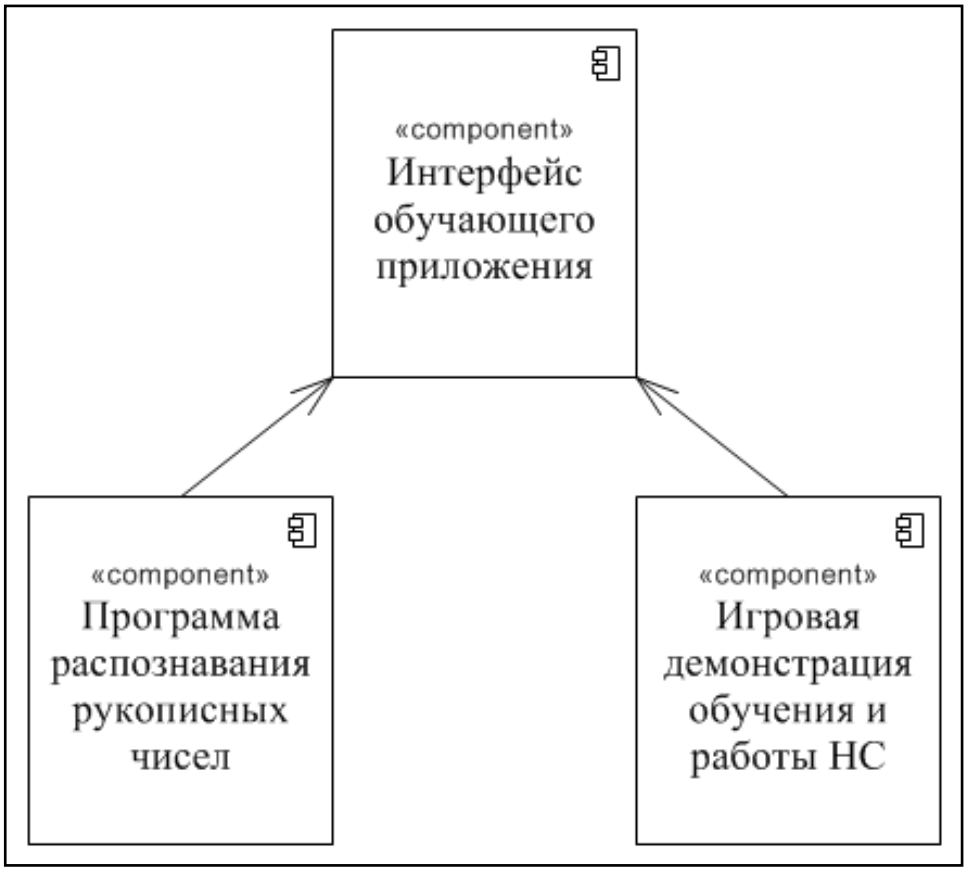


Рис. 6. Диаграмма компонентов обучающего приложения

2.5. Описание программы распознавания чисел

Входные данные: массив черно-белых изображений рукописных чисел размером 28*28 пикселей (рис. 7). Изображения были взяты из библиотеки MNIST [2].

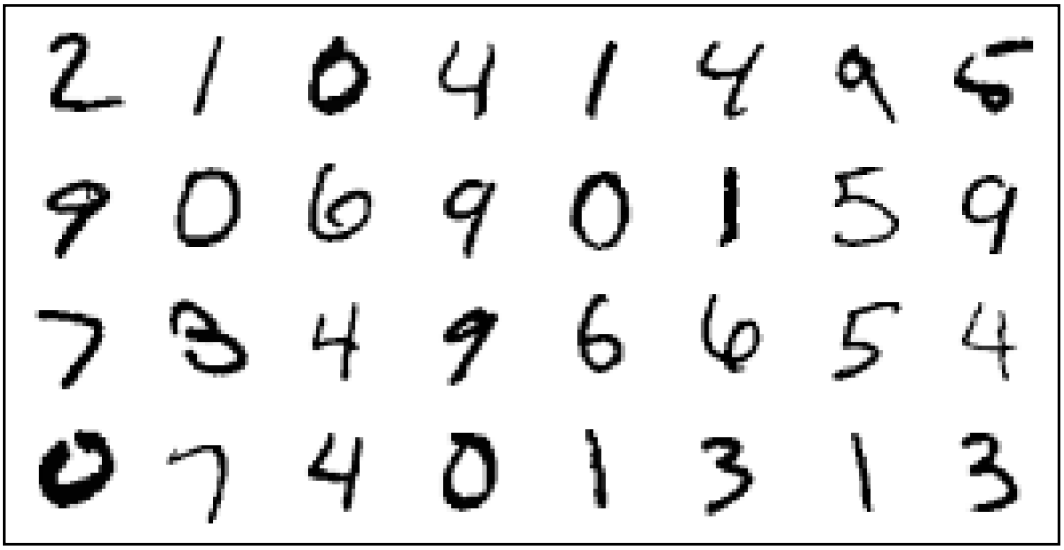


Рис. 7. Пример изображений из библиотеки MNIST

Выходные данные: результат распознавания каждого изображения в формате: «число» – «вероятность в процентах», значения весов нейронов выходного слоя.

На рис. 8 изображена диаграмма деятельности обучения нейронной сети. Данная диаграмма раскрывает алгоритм, по которому происходит обучение нейронной сети.

После запуска обучения сети, программа считывает обучающую выборку из файлов библиотеки MNIST. Далее в цикле преобразует каждое изображение в массив с числовыми значениями, в котором каждое число – значение пикселя изображения.

Затем программа производит распознавание изображения, высчитывает ошибку и меняет значения весов нейронов.

По завершении обучения сети программа переходит в режим распознавания изображений.

Нейронная сеть реализована на языке Python в среде программирования PyCharm [16] с использованием библиотеки Anaconda [14].

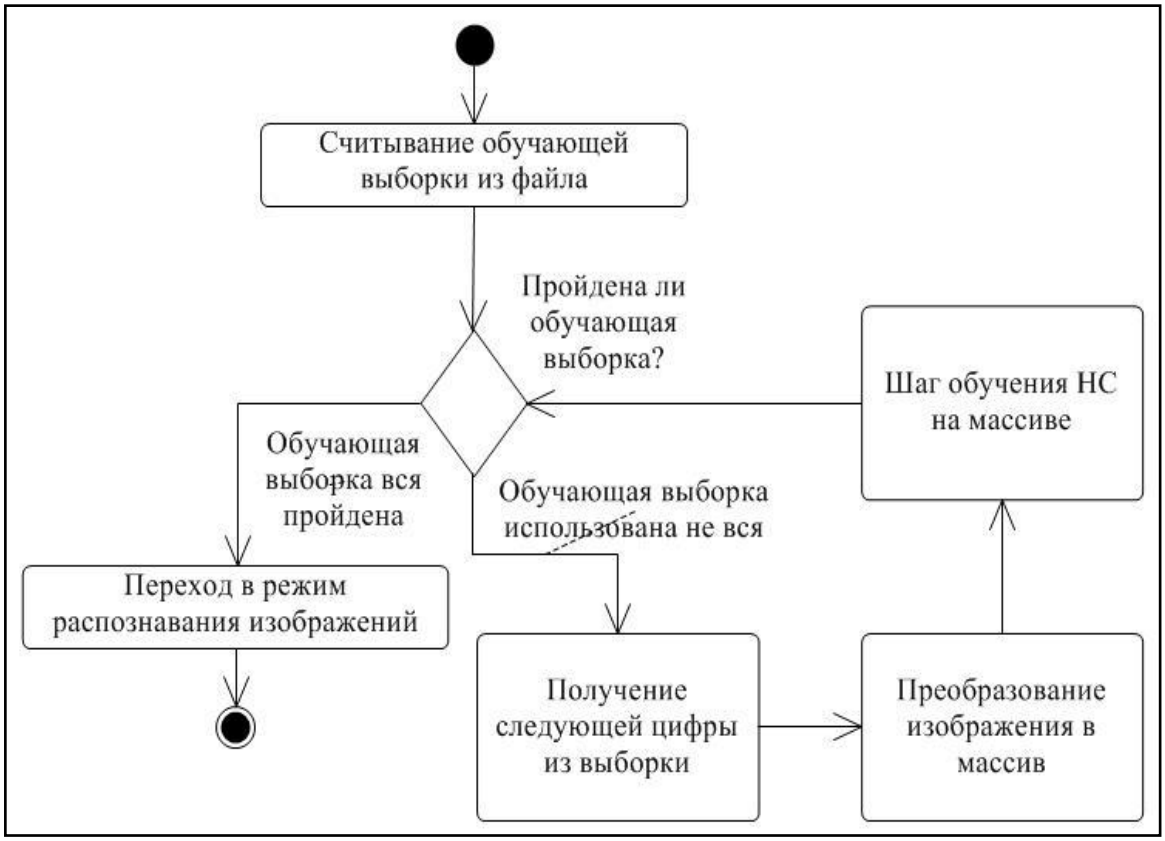


Рис. 8. Диаграмма деятельности обучения нейронной сети

Рис. 9. Вложенная диаграмма деятельности действия «Шаг обучения НС на массиве»

2.6. Описание игровой программы, демонстрирующей работу НС

Программа использует нейронную сеть для определения принадлежности цветного изображения одному из двух классов: «Съедобное» и «Несъедобное». Сеть обучается в ходе выполнения приложения: персонаж захватывает некоторый объект, пока он падает, сеть определяет класс объекта по его изображению и, в зависимости от результатов распознавания, персонаж решает, ловить объект или уклоняться от него. Если персонаж поймал несъедобный объект, то сеть переобучается, а у персонажа отнимается единица жизни. Если персонаж поймал съедобный объект, то единица жизни прибавляется. На рис. 10 изображена диаграмма деятельности игровой программы.

Входные данные нейронной сети: цветное изображение 150*150 пикселей. Примеры изображений приведены на рис. 11.

Выходные данные нейронной сети: класс изображения: «Съедобное» или «Несъедобное».

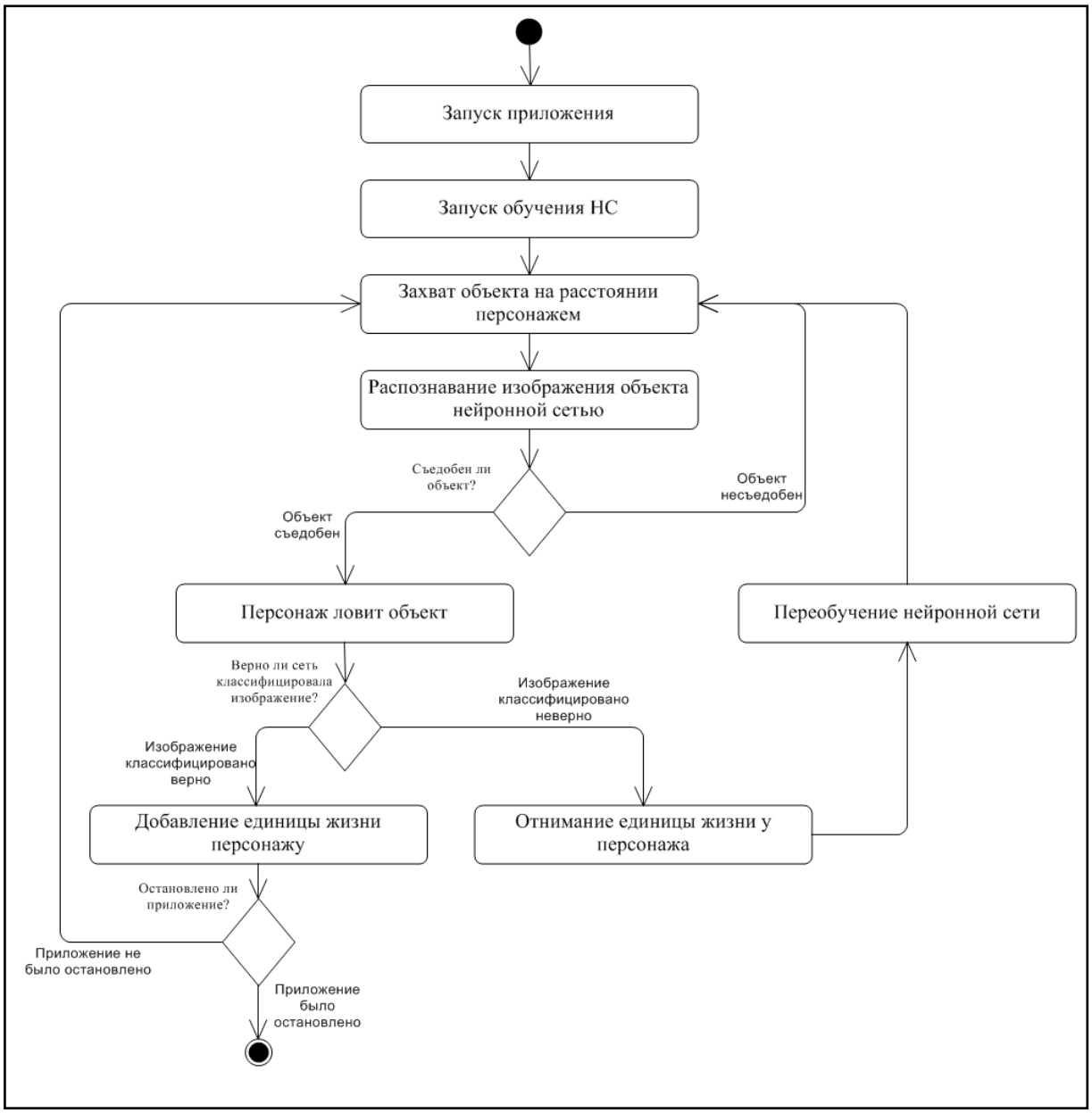


Рис. 10. Диаграмма деятельности игровой программы

Рис. 11. Изображения объектов

Распознавание изображения объекта нейронной сетью происходит по той же схеме, что и распознавание рукописных символов в п. 2.5.

3. РЕАЛИЗАЦИЯ

3.1. Разработка кода нейронной сети

Скрипт на языке Python запускает функцию, которая в теле цикла считывает изображение и класс изображения из библиотеки, производит обучение на полученной обучающей выборке (листинг 1). Обучение проходит в одну эпоху. Всего изображений в обучающей выборке десять тысяч. Столько же изображений находится и в библиотеке для распознавания.

После обучения запускаются функции распознавания изображений и вывода полученных результатов (листинг 2). Пример вывода приведен в листинге 3.

3.2. Реализация игрового демонстрационного приложения

3.2.1. Описание программы

Данное приложение знакомит пользователя с работой нейронной сети в процессе ее обучения. На вход нейронной сети подается цветное изображение 150*150 пикселей – цель, которую ловит персонаж, сеть должна распознать, изображено съедобное или нет. Если пойманное съедобно – персонажу (мышке) добавляется единица жизни, если несъедобно и персонаж не успел уклониться от объекта – единица жизни отнимается. Если жизней не остается, персонаж погибает.

Цель приложения–создать ИИ-робота, который сможет учиться оптимальной игре. В результате персонаж сможет спокойно ловить цель и пропускать ловушки. В наилучшем сценарии персонаж не умрет никогда.

Приложение написано на языке C# на платформе Unity [17].

3.2.2. Разработка кода программы

Персонаж движется под управлением обученной НС. Задача персонажа: ловить падающие объекты и набирать за них баллы, если пойманное съедобно, и терять баллы, если – несъедобно. Если количество единиц

жизни становится равным нулю, то персонаж погибает. Пример скрипта для управления персонажем приведен в листинге 4.

Для сравнения в приложении реализовано два персонажа, каждый из которых управляется собственной НС. Это наглядно демонстрирует то, что по факту одинаковые нейронные сети могут обучаться с разной скоростью и выдавать незначительные различия результатов обучения, которые со временем нивелируются.

Интерфейс приложения приведен на рисунке 12. Все графические объекты интерфейса, в том числе персонажи, фон и падающие объекты-цели были отрисованы вручную в растровом редакторе GIMP [15].

Рис.12. Интерфейс приложения

3.3. Компоновка программ в одно обучающее приложение

Для удобства пользователя обе программы выведены в интерактивную оболочку Jupyter Notebook [13]. Данная среда позволяет активно взаимодействовать как с готовой программой, так и с ее кодом.

Само обучающее приложение оформлено в виде текстовых блоков с вводной информацией по нейронным сетям, блока работы с кодом программы, распознающей числа, окна с интерактивным приложением, де-

монстрирующим нейронную сеть в процессе обучения и работы, и описанием приведенных программ. Структура приложения iPython Notebook приведена на рис. 13.

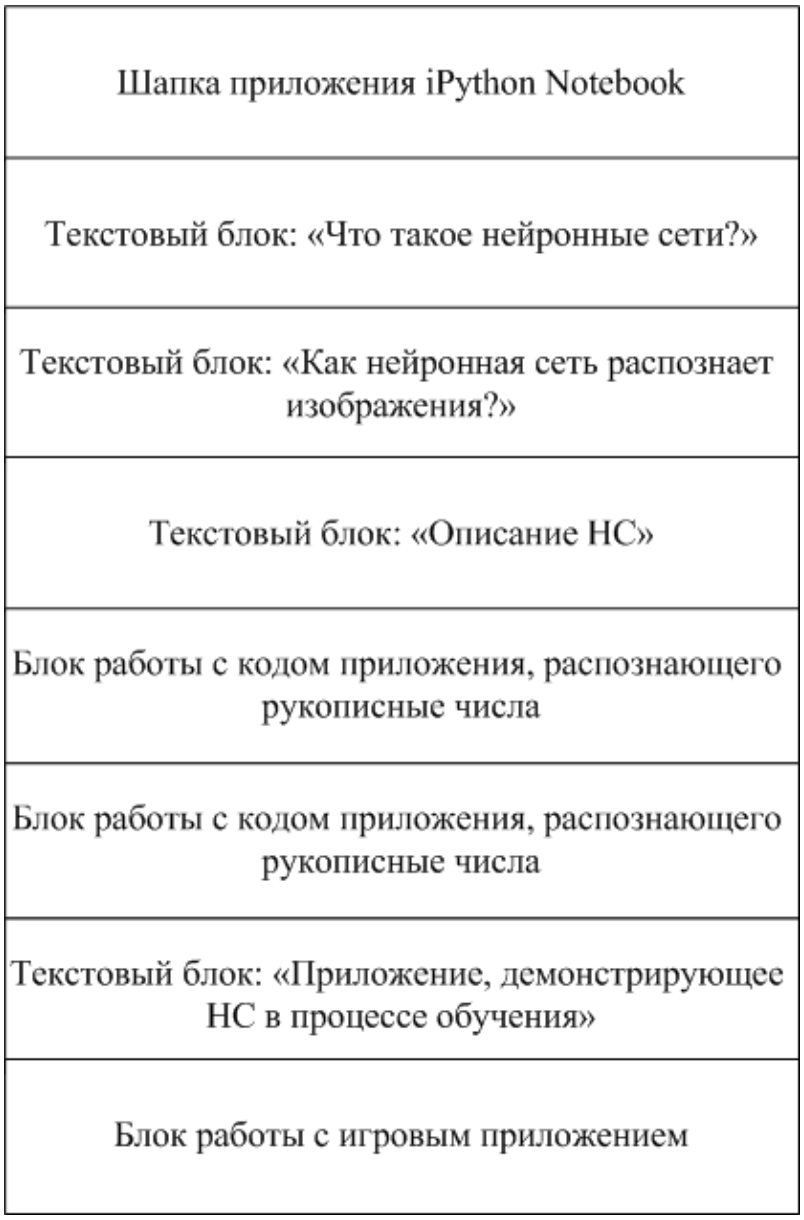


Рис. 13. Структура приложения

Шапка приложения iPython Notebook – это стандартная шапка платформы iPython Notebook с базовыми функциями редактирования и запуска кода и сохранения изменений.

Текстовые блоки «Что такое нейронная сеть?» и «Как нейронная сеть распознает изображения?» раскрывают основные понятия сетей, описыва-

ют, как они устроены, как происходит обучение и распознавание изображений.

Текстовый блок «Описание НС» описывает программу распознавания рукописных чисел. Следующий блок работы с кодом позволяет запустить данную программу, вывести в консоль ее функции, отредактировать их.

Текстовый блок «Приложение, демонстрирующее НС в процессе обучения» – это блок описания игрового приложения, приведенного в следующем блоке, демонстрирующим нейронную сеть в процессе обучения и работы.

Готовый интерфейс приложения приведен на рисунке 14.

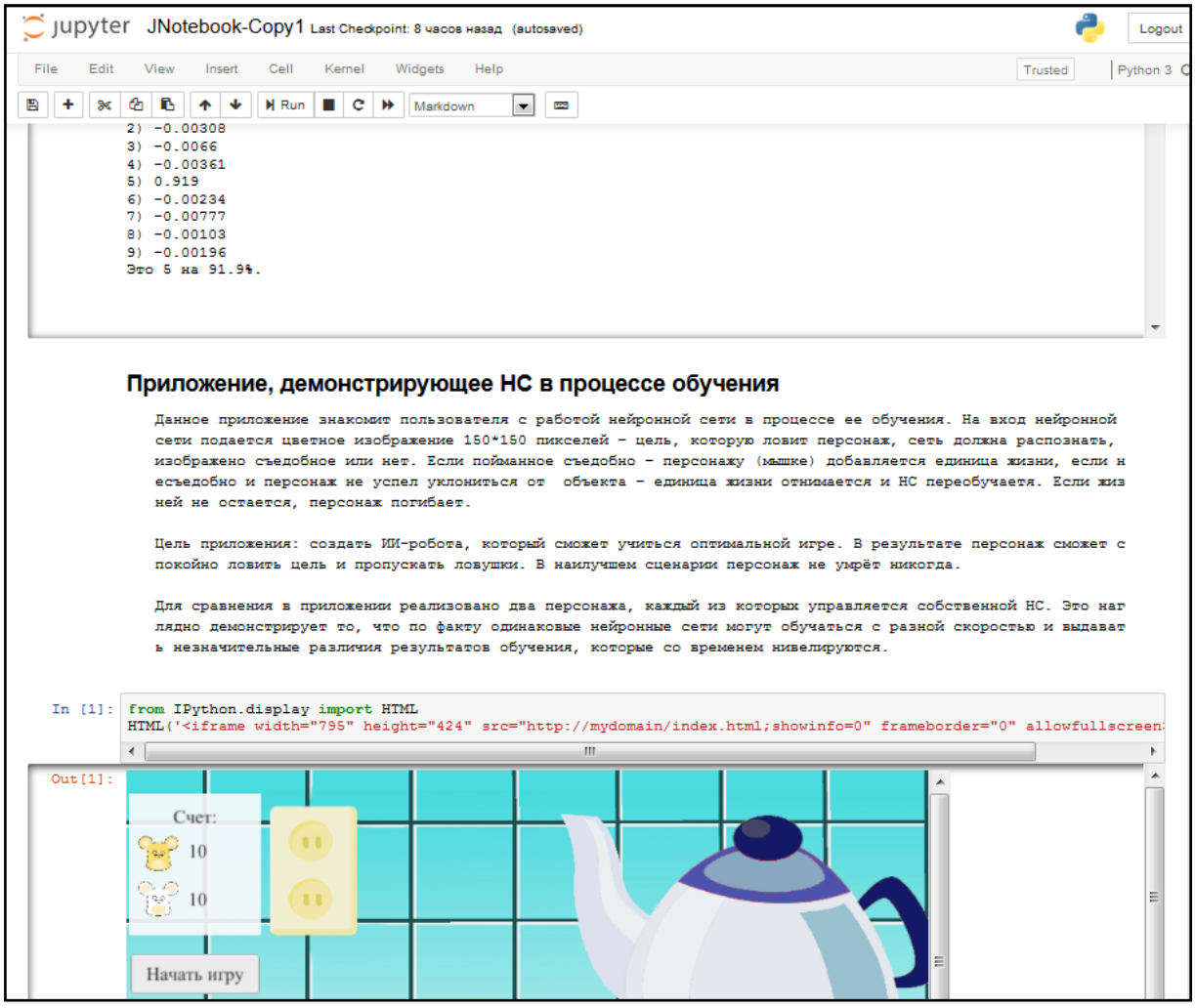


Рис. 14. Интерфейс приложения

4. ТЕСТИРОВАНИЕ

4.1. Функциональное тестирование

В ходе тестирования проводилась проверка корректной работы нейронных сетей обоих программ (программы распознавания чисел на языке Python и программы распознавания съедобного-несъедобного на C#). В обоих случаях на вход подавались массивы изображений с заранее известной классификацией. Массив черно-белых рукописных чисел был взят из библиотеки MNIST. Массив картинок «съедобное-несъедобное» был создан вручную в растровом редакторе Gimp.

Обе программы показали хорошие результаты: общее среднее количество правильно распознанных изображений – 98,19 %. Подсчет производился методом определения отношения количества правильно распознанных изображений к общему числу изображений в тестовой выборке, сначала в каждой программе по отдельности, затем вычислялось среднее между ними. Результаты тестирования приведены в таблицах 1 и 2.

Табл. 1. Результаты тестирования программы, распознающей рукописные числа

№	**Количество изображений в массиве**	**Процент правильно распознанных изображений**	**Тест пройден?**
1.	50	99,7	Да
2.	100	99,66	Да
3.	300	99,63	Да
4.	700	99,57	Да
5.	1000	99,5	Да
6.	2500	99,3	Да
7.	5000	98,9	Да
8.	7500	98,72	Да
9.	10000	98,5	Да
Среднее значение			99,3

Табл. 2. Результаты тестирования программы, распознающей съедобное и несъедобное

№	Количество изображений в массиве	Процент правильно распознанных изображений	Тест пройден?
1.	5	97,3	Да
2.	7	97,3	Да
3.	10	97,1	Да
4.	15	96,9	Да
5.	20	96,8	Да
Среднее значение			97,08

4.2. Интеграционное тестирование

В ходе денного тестирования готовое приложение было запущено в нескольких браузерах: Chrome, Mozilla, Yandex. Результаты тестирования приведены в таблице 3.

Табл. 3. Результаты тестирования программы в различных браузерах

№	Браузер	Результаты тестирования	Тест пройден?
1.	Google Chrome	Приложение запустилось и работает без ошибок	Да
2.	Mozilla FireFox	Приложение запустилось и работает без ошибок	Да
3.	Yandex Browser	Приложение запустилось и работает без ошибок	Да

Вывод по тестированию

В ходе функционального тестирования был произведен подсчет количества верного распознавания изображений и выведен средний показатель точности нейронных сетей: 98,19 %.

Интеграционное тестирование показало, что вне зависимости от браузера все кнопки приложения активны, программы запускаются и работают корректно, интерфейс не съезжает, текстовые блоки так же отображаются корректно.

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломной работы было разработано две программы, одна из которых предназначена для изучения программной составляющей нейронных сетей (приложение, распознающее рукописные числа). Другая (приложение, распознающее съедобные и несъедобные изображения) – для знакомства пользователя в игровом формате с обучением НС и ее работой в режиме реального времени. Обе программы являются частью одного игрового приложения для обучения основам работы нейронных сетей.

Для выполнения поставленной цели были решены следующие задачи:

1) произведен подбор литературы, необходимой для разработки программы;
2) выполнен анализ существующих решений;
3) определены ключевые требования программы;
4) программа была спроектирована, реализована и протестирована.

СПИСОК ЛИТЕРАТУРЫ

1. Machine Learning Algorithm for Flappy Bird using Neural Network and Genetic Algorithm. [Электронный ресурс] URL: https://www.askforgametask.com/ tutorial/machine-learning-algorithm-flappy-bird/ (дата обращения: 04.09.2018).
2. MNIST база данных рукописных цифр. [Электронный ресурс] URL: http://yann.lecun.com/exdb/mnist/ (дата обращения: 09.04.2018).
3. TensorFlow playground. [Электронный ресурс] URL: https://playground.tensorflow.org (дата обращения: 04.03.2018).
4. Гуляева А. Список курсов по нейронным сетям. [Электронный ресурс] URL: https://apptractor.ru/learn/13-besplatnyih-kursov-po-neyronnyim-setyam-i-mashinnomu-obucheniyu.html (дата обращения: 24.06.2018).
5. Игры, которые учат программированию. [Электронный ресурс] URL: https://habrahabr.ru/post/273003/ (дата обращения: 23.05.2018).
6. Как добиться того, чтобы обучение в играх не раздражало. [Электронный ресурс] URL: https://habrahabr.ru/post/330948/ (дата обращения: 23.05.2018).
7. Карпенко О. Необычные игры, которые учат программированию и логике. [Электронный ресурс] URL: http://ain.ua/2015/06/26/10-neobychnyx-igr-kotorye-uchat-programmirovaniyu-i-logike (дата обращения: 23.05.2018).
8. Контсаренко Ф. Опыт команды Google Primer: Создание интерфейса обучающего приложения. [Электронный ресурс] URL: https://vc.ru/11659-ux-learn (дата обращения: 12.01.2019).
9. Курс Deep Learning. [Электронный ресурс] URL: https://www.udacity.com/course/deep-learning--ud730 (дата обращения: 24.12.2018).

10. Михайленко Т.М. Игровые технологии как вид педагогических технологий. // Педагогика: традиции и инновации. Материалы Междунар. науч. конф. (г. Челябинск, октябрь 2011 г.). Т. I. – Челябинск: Два комсомольца, 2011. – С. 140-146.

11. Москалев Н.С. Виды архитектур нейронных сетей. // Молодой ученый. – 2016. – № 29. – С. 30-34. [Электронный ресурс] URL: https://moluch.ru/archive/133/37121/ (дата обращения: 14.01.2019).

12. Обзор курсов по Deep Learning. [Электронный ресурс] URL: https://habrahabr.ru/company/newprolab/blog/311812/ (дата обращения: 23.05.2018).

13. Официальный сайт разработчиков IPython. [Электронный ресурс] URL: https://jupyter.org/ (дата обращения: 01.12.2018).

14. Официальный сайт разработчиков дополнения Anaconda. [Электронный ресурс] URL: https://www.anaconda.com/ (дата обращения: 12.09.2018).

15. Официальный сайт разработчиков растрового редактора GIMP. [Электронный ресурс] URL: https://www.gimp.org/ (дата обращения: 15.03.2019).

16. Официальный сайт разработчиков среды программирования PyCharm. [Электронный ресурс] URL: https://www.jetbrains.com/pycharm/ (дата обращения: 12.09.2018).

17. Официальный сайт разработчиков среды программирования Unity. [Электронный ресурс] URL: https://unity.com/ (дата обращения: 10.04.2015).

18. Распознавание оптических образов (символов) с помощью однослойного персептрона. Методические указания по дисциплинам «Системы искусственного интеллекта», «Представление и обработка знаний». [Электронный ресурс] URL: http://www.myshared.ru/slide/725725/ (дата обращения: 10.05.2018).

19. Рашид Т. Создай свою нейросеть. / Т. Рашид ; пер. с англ. Радько П. – М. : Вильямс, 2017. – 272 с.

20. Фролова Ю. 10 игр для изучения программирования [Электронный ресурс] URL: https://geekbrains.ru/posts/10_games (дата обращения: 23.05.2018).

ПРИЛОЖЕНИЕ

Листинг 1. Функция формирования обучающей пары (Program.py)

```
    def learning(self) ->None:
        print("Обучение.", flush=True)
        pixelsBuffer = []
        for r in range(28):  # строки
            pixelsBuffer.append([])  # создаем пустую строку
                for c in range(28):  # в каждой строке columns
                    pixelsBuffer[r].append(0)
        epoch = 0
        while epoch < 1:
            print(str(epoch + 1) + " эпоха. Прогресс обучения:")
            ifsLabels=open("E:/Project/pyCode/proj/t10k-labels.idx1-
ubyte", "rb")
            ifsImages=open("E:/Project/pyCode/proj/t10k-images.idx3-
ubyte", "rb")
            brImages = BinaryStream(ifsImages)
            brLabels = BinaryStream(ifsLabels)
            magic1 = brImages.readInt32()
            numImages = brImages.readInt32()
            numRows = brImages.readInt32()
            numCols = brImages.readInt32()
            magic2 = brLabels.readInt32()
            numLabels = brLabels.readInt32()
            #print("Внесение обучающей выборки в базу обучения:")
            di = 0
            while di < 10000:
                i = 0
                while i< 28:
                    j = 0
                    while j < 28:
                    pixelsBuffer[i][j] =
int.from_bytes(brImages.readByte(), byteorder='big', signed=True)
                    j += 1
                i += 1
                lbl = int.from_bytes(brLabels.readByte(), byteord-
er='big', signed=True)
                Program.__net.study(pixelsBuffer, lbl)
                print('\r' + str((di + 1) / 100) + ' / 100%', end=' ')
                di += 1
                print()
                epoch += 1
```

```
        ifsImages.close()
        ifsLabels.close()
        print()
```

Листинг 2. Функции распознавания изображения и вывода результатов (Program.py)

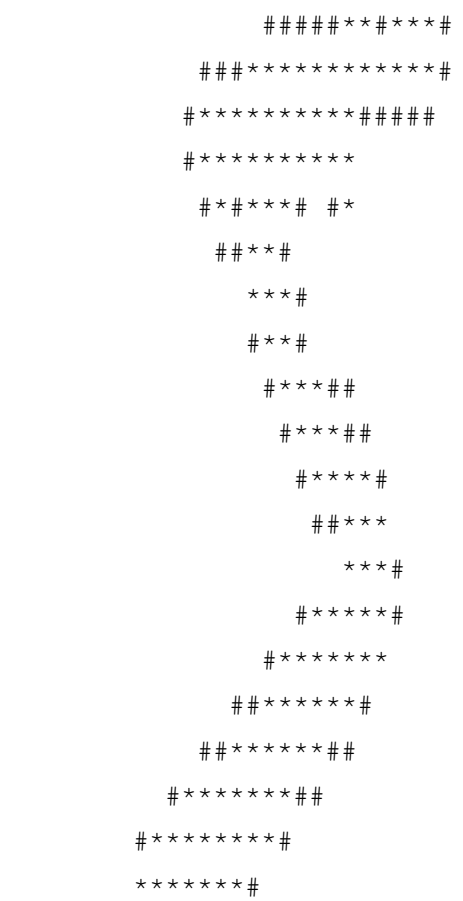
```
    def recognition(self, imageNumber : int ) -> None:
        print("Распознавание...", flush=True)
        ifsLabels = open("E:/Project/pyCode/proj/train-labels.idx1-
ubyte", "rb")
        ifsImages = open("E:/Project/pyCode/proj/train-images.idx3-
ubyte", "rb")
        brImages = BinaryStream(ifsImages)
        brLabels = BinaryStream(ifsLabels)
        magic1 = brImages.readInt32()
        numImages = brImages.readInt32()
        numRows = brImages.readInt32()
        numCols = brImages.readInt32()
        magic2 = brLabels.readInt32()
        numLabels = brLabels.readInt32()
        pixels = []
        for r in range(28):  # строки
            pixels.append([])  # создаем пустую строку
            for c in range(28):  # в каждой строке columns
                pixels[r].append(0)
        di = 0
            while di < 10000:
            i = 0
                while i< 28:
                j = 0
                while j < 28:
                    pixels[i][j] = int.from_bytes(brImages.readByte(),
byteorder='big', signed=True)
                    j += 1
                i += 1
            lbl = int.from_bytes(brLabels.readByte(), byteorder='big',
signed=True)
            self.__test_images[di] = DigitImage(pixels, lbl)
            print(str(self.__outAns(self.__test_images[di])), flush=True)
            print(self.__str(self.__test_images[di]), flush=True)
            di += 1
```

```
        ifsImages.close()
        ifsLabels.close()
        print("End", flush=True)
    def __outAns(self, dimage : 'DigitImage') -> str:
        ansnum = Program.__net.getAnswer(dimage.pixels)
        cht = ""
        while not (ansnum == dimage.label):
            Program.__net.study(dimage.pixels, dimage.label)
            ansnum = Program.__net.getAnswer(dimage.pixels)
            output = Program.__net.handle(dimage.pixels)
            n = Utils.newArray(len(output), 0)
            i = 0
            while i<len(output):
                a = output[i]
                if (a < 0):
                cht = str(a)[0:0+4]
                else:
                cht = str(a)[0:0+3]
                if (i == ansnum):
                    n[i] = ((1) - math.fabs(n[i] * (10)))
                print(str(i) + ") " + str(n[i]))
                i += 1
        ans = ("Это " + (str(ansnum)) + " на ") + str(n[ansnum] * (100))
+ "%."
        return ans
```

Листинг 3. Пример вывода программы (Program.py)

```
0) -0.0029
1) -0.00123
2) -0.00308
3) -0.0066
4) -0.00361
5) 0.919
6) -0.00234
7) -0.00777
8) -0.00103
9) -0.00196


Это 5 на 91.9 %
```



Листинг 4. Класс персонажа (PlayerController.cs)

```
public class PlayerController : MonoBehaviour
{
    Public float speed = 20f; //Скорость движения персонажа
    private Rigidbody2D rb;
    public Text text;
    public Text catchText;
    public float score;
    Program program = new Program();

    void Start()
    {
        rb = GetComponent<Rigidbody2D>();
        score = 0;
        program.Learning(); //  Запуск обучения нейронной сети
    }

    void Update() //Функция, реализующая движение персонажа
    {
```

```
        text.text = ((int)score).ToString(); //вывод количества жизней
        float moveX = Input.GetAxis("Horizontal");
        //если персонаж не вылетел вбок
        if (rb.position.x> -10 &&rb.position.x< 10){
            if (catchText.text != ""){
                Vector2 vector2 = new Vec-
tor2(float.Parse(catchText.text),0f);
                moveX = program.GetMoveX(vector2.x -
rb.position.x,Time.deltaTime);
                rb.MovePosition(rb.position + (Vector2.right) * moveX *
speed * Time.deltaTime);
            }
        }
        else {
            if (rb.position.x<= -10) rb.MovePosition(rb.position + (Vec-
tor2.right) * 0.1f * speed * Time.deltaTime);
            if (rb.position.x>= 10) rb.MovePosition(rb.position + (Vec-
tor2.left) * 0.1f * speed * Time.deltaTime);
        }
    }
    void OnTriggerEnter2D(Collider2D other)
    {   //Получаем изображение цели
        Texture2D sprite = oth-
er.gameObject.GetComponent<SpriteRenderer>().sprite.texture;
        byte[] tex = sprite.EncodeToPNG();
        //Распознаем изображение
        int ans = program.Recognition(tex);
        // Если съедобно, то разрушаем объект и добавляем жизнь
        if (other.gameObject.CompareTag("Platform")){
            Destroy(other.gameObject);
            score++;
        }
        // Если не съедобно, то разрушаем объект, отнимаем жизнь и переобу-
чаем НС
        if (other.gameObject.CompareTag("Bane")){
            Destroy(other.gameObject);
            score--;
            program.ReLearning(tex, 0);
        }
    }
}
```