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**

|  |
| --- |
| ДОПУСТИТЬ К ЗАЩИТЕ  Заведующий кафедрой, д.ф.-м.н., профессор  \_\_\_\_\_\_\_\_\_\_ Л.Б. Соколинский  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2021 г. |

**Разработка голосового помощника для ОС Windows на основе нейросетвых технологий**

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

ЮУрГУ – 09.03.04.2021.308-278.ВКР

|  |  |
| --- | --- |
|  | Научный руководитель,  к.ф.-м.н., доцент кафедры CП \_\_\_\_\_\_\_\_\_\_ С.А. Иванов  Автор работы, студент группы КЭ-402  \_\_\_\_\_\_\_\_\_\_ А.А. Владимиров  Ученый секретарь  (нормоконтролер)  \_\_\_\_\_\_\_\_\_\_\_\_\_ И.Д. Володченко  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2021 г. |

Челябинск, 2021 г.

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

08.02.2021 г.

**ЗАДАНИЕ**

**на выполнение выпускной квалификационной работы бакалавра**

студенту группы КЭ-402

Владимирову Артёму Александровичу,

обучающемуся по направлению

02.03.02 «Фундаментальная информатика и информационные технологии»

1. **Тема работы** (утверждена приказом ректора от 26.04.2021 г. № 714-13/12)

Разработка голосового помощника для ОС Windows на основе нейросетвых технологий.

1. **Срок сдачи студентом законченной работы:** 05.06.2021 г.
2. **Исходные данные к работе**
3. Хайкин С. Нейронные сети: полный курс. 2-е изд. Москва: ООО "И.Д. Вильямс", 2017. 1104 с.
4. Russian Open Speech To Text (STT/ASR) Dataset [Электронный ресурс] URL: https://github.com/snakers4/open\_stt/ (дата обращения: 28.03.2021).
5. **Перечень подлежащих разработке вопросов**
6. Провести обзор аналогов и научной литературы.
7. Подготовить обучающую и тестовую выборку аудио файлов.
8. Программно реализовать и обучить нейронную сеть.
9. Разработать и протестировать приложение.
10. **Дата выдачи задания:** 08.02.2021 г.

**Научный руководитель,**

к.ф.-м.н., доцент кафедры CП С.А. Иванов

**Задание принял к исполнению** А.А. Владимиров

Оглавление

[ВВЕДЕНИЕ 5](#_Toc72665054)

[1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 7](#_Toc72665055)

[1.1. Cortana 7](#_Toc72665056)

[1.2. Обзор существующих решений для распознавания речи 8](#_Toc72665057)

[1.3. Обзор готовых решений по созданию нейронных сетей и обработки звука 10](#_Toc72665058)

[2. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 12](#_Toc72665059)

[2.1. Представление звуковой информации в компьютере 12](#_Toc72665060)

[2.2. Предварительная обработка данных 13](#_Toc72665061)

[2.3. Сверточная нейронная сеть 16](#_Toc72665062)

[3. ПРОЕКТИРОВАНИЕ 19](#_Toc72665063)

[3.1. Функциональные требования 19](#_Toc72665064)

[3.2. Нефункциональные требования 19](#_Toc72665065)

[3.3. Диаграмма вариантов использования 20](#_Toc72665066)

[3.4. Топология нейронной сети 21](#_Toc72665067)

[3.5. Разработка модуля выполнения команд 22](#_Toc72665068)

[4. РЕАЛИЗАЦИЯ 25](#_Toc72665069)

[4.1. Программные средства реализации 25](#_Toc72665070)

[4.2. Формирование обучающей выборки 25](#_Toc72665071)

[4.3. Реализация нейронной сети 27](#_Toc72665072)

[4.4. Реализация распознавания слов 28](#_Toc72665073)

[4.5. Реализация модуля выполнения команд 29](#_Toc72665074)

[4.6. Реализация графического интерфейса 31](#_Toc72665075)

[5. ТЕСТИРОВАНИЕ 32](#_Toc72665076)

[5.1. Тестирование нейронной сети 32](#_Toc72665077)

[5.2. Функциональное тестирование 32](#_Toc72665078)

[ЗАКЛЮЧЕНИЕ 34](#_Toc72665079)

[ЛИТЕРАТУРА 35](#_Toc72665080)

[ПРИЛОЖЕНИЕ 38](#_Toc72665081)

[Приложение А. Код класса нейросети 38](#_Toc72665082)

[Приложение Б. Код модуля распознавания слов 39](#_Toc72665083)

введение

**Актуальность темы**

В последние годы популярность искусственных нейронных сетей [1] значительно возросла, они используются для решения множества задач, например: распознавание образов – символов текста, изображений, образцов звуков и т.д.; классификация – распределение данных по параметрам; принятие решений и управление; кластеризация – разбиение множества входных сигналов на классы; прогнозирование – предсказание будущего значения некой последовательности на основе нескольких предыдущих значений. Основными достоинствами нейронных сетей являются: решение задач в условиях неопределенности, устойчивость к шумам во входных данных, гибкость структуры нейронных сетей.

В данной работе искусственные нейронные сети будут использованы для распознавания образов, в частности, для распознавания речи. Технология распознавания речи в настоящее время активно набирает популярность. Она уже есть в телефонах, игровых консолях и в смарт-часах, даже дома автоматизируются с помощью речи. Использование распознавания речи в операционной системе Windows позволит упростить работу с некоторыми ее функциями.

**Цель и задачи исследования**

Целью представленной работы является разработка приложения на основе распознавания речи [3-5] и искусственных нейронных сетей для упрощения использования некоторых функций ОС Windows за счет применения голосовых команд. Программа будет обрабатывать поток аудио данных, поступающих с микрофона компьютера и выполнять заданные команды. Для реализации поставленной цели потребуется выполнить следующие подзадачи:

1. провести обзор аналогов и научной литературы;
2. подготовить обучающую и тестовую выборку аудио файлов;
3. программно реализовать и обучить нейронную сеть;
4. разработать и протестировать приложение.

**Структура и объем работы**

Работа состоит из введения, пяти глав, заключения, списка литературы и приложений. Объем работы составляет 40 страниц, объем библиографии – 30 источников, объем приложений – 3 страницы.

**Содержание работы**

В первой главе проводится обзор аналогичных решений, а также анализ существующих библиотек для работы с нейронными сетями и обработки звука.

Во второй главе описаны способы представления звуковой информации в компьютере, процесс предварительной обработки аудио данных для лучшего обучения нейросети, а также приводятся теоретические сведения о строении сверточных нейронных сетей.

В третьей главе описаны функциональные и нефункциональные требования к системе, представлены диаграммы вариантов использования и описана топология нейросети.

В четвертой главе представлены этапы создания обучающей выборки, программная реализация предобработки аудио данных, модели нейронной сети, модуля распознавания речи и выполнения команд.

В пятой главе приведены результаты тестирования программы и нейронной сети.

В заключении описаны полученные в ходе выполнения работы результаты.

1. Анализ предметной области
   1. Cortana

Кортана (англ. *Cortana*) [6] – виртуальная голосовая помощница с элементами искусственного интеллекта от Microsoft для Windows Phone 8.1, Microsoft Band, Windows 10, Xbox One.

Впервые была продемонстрирована во время конференции Build в Сан-Франциско 2 апреля 2014 года. Кортана была названа в честь героини серии компьютерных игр Halo – голос помощницы в версии для американского рынка принадлежит Джен Тейлор, которая также озвучивала Кортану в оригинальной игре.

Персональная помощница Кортана призвана предугадывать потребности пользователя. При желании ей можно дать доступ к личным данным, таким как электронная почта, адресная книга, история поисков в сети и т. п. – все эти данные она будет использовать для упреждения нужд пользователя. Кортана заменит стандартную поисковую систему и будет вызываться нажатием кнопки «Поиск». Нужный запрос можно как напечатать вручную, так и задать голосом. Необходимую информацию она будет находить, опираясь на результаты поиска в системе Bing, Foursquare и среди личных файлов пользователя. Также виртуальный ассистент не лишена чувства юмора: она может поддерживать беседу, петь и рассказывать анекдоты. Она заранее напомнит о запланированной встрече, дне рождения друга и других важных событиях. Интерфейс имеет гибкие настройки конфиденциальности, позволяющие пользователю самому определять, какого рода информацию предоставлять виртуальному ассистенту.

Голосовая помощница Кортана интегрирована в Windows 10. Она не выступает в роли отдельного приложения, а интегрирована в поиск Windows 10.

Кортана доступна на английском, немецком, французском, итальянском, испанском, португальском, японском и китайском языках.

* 1. Обзор существующих решений для распознавания речи

**Yandex SpeechKit [7]**

Сервис Yandex SpeechKit позволяет разработчикам приложений использовать речевые технологии Яндекса: распознавание речи (Speech-to-Text) и синтез речи (Text-to-Speech). Для доступа к SpeechKit используется HTTP API.

Сервис позволяет распознавать речь на языках:

* русский;
* английский;
* турецкий.

Есть три способа распознавания.

* + 1. Распознавание коротких аудио. Подходит для распознавания одноканального аудио небольшого размера.
    2. Потоковый режим для распознавания коротких аудио. Позволяет в рамках одного соединения отправлять аудио фрагменты и получать результаты, в том числе промежуточные результаты распознавания.
    3. Распознавание длинных аудио. Позволяет распознавать длинные многоканальные аудиозаписи, но скорость ответа может быть ниже.

Распознавание аудио происходит в три этапа.

* + 1. Выделяются слова. Обычно существует несколько гипотез распознанного слова.
    2. Гипотезы проверяются с помощью языковой модели. Модель проверяет, насколько согласуется новое слово со словами, распознанными ранее.
    3. Обрабатывается распознанный текст – числительные преобразуются в цифры, расставляются некоторые знаки препинания (например, дефисы) и т. д. Этот преобразованный текст и является финальным результатом распознавания, который отправляется в теле ответа.

К 2020 году синтез и распознавание речи стали самым востребованным ML-сервисом на платформе Yandex.Cloud. С начала года объем потребления SpeechKit вырос на 120%. Количество активных проектов превысило 500. В России уже сформировалась экосистема разработчиков и интеграторов решений, которые по заказу компаний из различных сфер создают и внедряют голосовых роботов для помощи в обработке входящих и исходящих звонков, системы голосового управления в приложениях и терминалах обслуживания клиентов, решения по анализу эффективности бизнес-коммуникаций. Сегодня это более 20 компаний, большинство из которых – постоянные партнеры платформы Yandex.Cloud.

**Speereo Voice Assistant [8]**

Это программно-аппаратное решение, которое позволяет использовать речевой интерфейс «поверх» любого ПО. SVA распознает речевые команды и запускает соответствующие макросы, эмулируя клавиатуру и мышь. Время распознавания и начала исполнения макроса – меньше секунды. SVA использует собственный речевой движок компании Speereo, который показывает очень высокую точность на таких задачах (98-99,9%).

По словам разработчиков приложение обладает рядом преимуществ.

* + 1. Персонализация. Можно настроить все тексты команд как угодно.
    2. Скорость. Распознавание происходит на компьютере пользователя, без внешних серверов.
    3. Концентрация. Можно сконцентрироваться на своей работе, не отвлекайтесь на системы меню и экранные кнопки.
    4. Простор. Можно убрать интерфейсные панели из своего рабочего приложения и больше места оставить под рабочую область.
    5. Свобода. Можно работать стоя, использовать проекторы и системы vr.
    6. Контекст. Набор команд переключается автоматически в зависимости от активного приложения.
  1. Обзор готовых решений по созданию нейронных сетей и обработки звука

**PyTorch [9]**

Современная библиотека глубокого обучения, развивающаяся под крылом Facebook. PyTorch является аналогом фреймворка Torch7 для языка Python. Разработка его началась в недрах Facebook еще в 2012 году, всего на год позже появления самого Torch7, но открытым и доступным широкой публике PyTorch стал лишь в 2017 году. Фреймворк очень быстро набирает популярность и привлекает внимание все большего числа исследователей. Отличительной возможностью является возможность использовать видеокарты для вычислений.

**Scikit-Learn [10]**

Scikit-learn - один из наиболее широко используемых пакетов Python для Data Science и Machine Learning, основан на библиотеках NumPy и SciPy. Он позволяет выполнять множество операций и предоставляет множество алгоритмов. Scikit-learn также предлагает отличную документацию о своих классах, методах и функциях, а также описание используемых алгоритмов.

Scikit-Learn поддерживает:

* предварительную обработку данных;
* уменьшение размерности;
* выбор модели;
* регрессии;
* классификации;
* кластерный анализ.

Он также предоставляет несколько наборов данных, которые можно использовать для тестирования собственных моделей.

Scikit-learn не реализует все, что связано с машинным обучением. Например, он не имеет комплексной поддержки для:

* нейронных сетей;
* самоорганизующихся карт (сетей Кохонена);
* обучения ассоциативным правилам;
* обучения с подкреплением (reinforcement learning).

Scikit-learn – это пакет с открытым исходным кодом. Как и большинство материалов из экосистемы Python, он бесплатный даже для коммерческого использования.

**Librosa [11]**

Это пакет Python для анализа музыки и аудио. Он предоставляет строительные блоки для создания структур, которые помогают получать информацию о музыке. Librosa может работать с любыми звуковыми сигналами, но ориентирована в основном именно на музыку. Она позволяет создать полноценную систему извлечения музыкальной информации. Модуль прекрасно документирован, кроме того, существует множество руководств по использованию.

**SciPy [12]**

Это [библиотека Python](https://pythonru.com/biblioteki) с открытым исходным кодом, предназначенная для решения научных и математических проблем. Она построена на базе NumPy и позволяет управлять данными, а также визуализировать их с помощью разных высокоуровневых команд. В том числе она содержит модуль signal для работы с аудио файлами.

**Выводы по первой главе**

В этой главе была рассмотрена часть аналогов, реализующих похожие функции. Стоит отметить большое разнообразие программных библиотек для задач машинного обучения с использованием нейронных сетей. Использование подобных библиотек может значительно облегчить задачу реализации. Совокупность озвученных фактов позволяет сделать вывод о актуальности решаемой задачи.

1. Теоретическая часть
   1. Представление звуковой информации в компьютере

Звук представляет собой непрерывный сигнал – звуковую волну с меняющейся амплитудой и частотой [13]. Чем больше амплитуда сигнала, тем он громче для человека. Чем больше частота сигнала, тем выше тон.

Частота звуковой волны выражается числом колебаний в секунду и измеряется в герцах (Гц). Человеческое ухо способно воспринимать звуки в диапазоне от 20 Гц до 20 кГц, который называют звуковым.

Процесс перевода звуковых сигналов от непрерывной формы представления к дискретной, цифровой форме называют оцифровкой. Важной характеристикой при кодировании звука является частота дискретизации – это количество измерений громкости звука за одну секунду. Количество измерений может лежать в диапазоне от 8 кГц до 48 кГц (от частоты радиотрансляции до частоты, соответствующей качеству звучания музыкальных носителей).

Существуют различные методы кодирования звуковой информации двоичным кодом, среди которых можно выделить два основных направления: метод FM и метод Wave-Table.

**Метод FM** (Frequency Modulation) основан на том, что теоретически любой сложный звук можно разложить на последовательность простейших гармонических сигналов разных частот, каждый из которых представляет собой правильную синусоиду, и, следовательно, может быть описан кодом. Разложение звуковых сигналов в гармонические ряды и представление в виде дискретных цифровых сигналов выполняют специальные устройства – **аналогово-цифровые преобразователи** (АЦП), пример такого преобразования представлен на рисунке 1: **a**– звуковой сигнал на входе АЦП; **б**– дискретный сигнал на выходе **АЦП**.

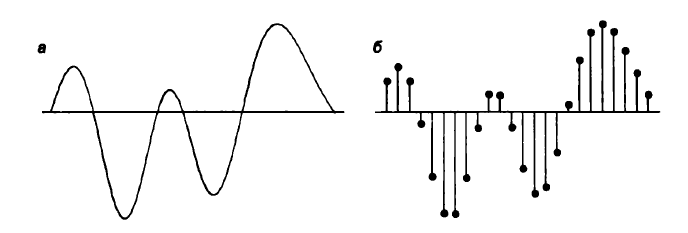


Рисунок 1 – Преобразование звукового сигнала в дискретный сигнал

**Таблично-волновой метод** (Wave-Table) основан на том, что в заранее подготовленных таблицах хранятся образцы звуков окружающего мира, музыкальных инструментов и т. д. Числовые коды выражают высоту тона, продолжительность и интенсивность звука и прочие параметры, характеризующие особенности звука. Поскольку в качестве образцов используются «реальные» звуки, качество звука, полученного в результате синтеза, получается очень высоким и приближается к качеству звучания реальных музыкальных инструментов.

* 1. Предварительная обработка данных

В данной работе будут использоваться аудио файлы в формате WAV с чистотой дискретизации 16 кГц. Каждый из которых можно представить в виде массива чисел, которые отражают значения амплитуды каждую 1/16000 секунды. Для облегчения обучения нейросети нужно провести предварительную обработку этих данных.

Для начала сгруппируем отсчеты во фрагменты по 20 миллисекунд. В каждом из таких фрагментов получится по 320 значений, пример такого фрагмента изображен на рисунке 2.

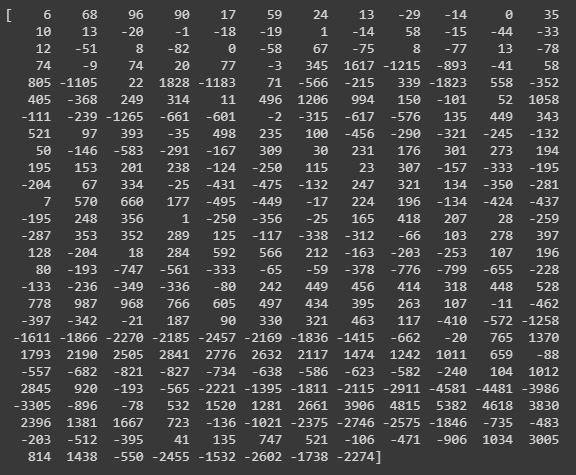


Рисунок 2 – Пример фрагмента аудио файла

Эта запись длится всего 1/50 секунды. Но даже она представляет собой сложную смесь различных частот звука. Есть несколько низких звуков, есть среднечастотные звуки и даже некоторые высокие звуки. Все эти частоты смешиваются вместе – и получается звук человеческой речи.

Чтобы упростить обработку этих данных для нейронной сети, разложим сложную звуковую волну на ее составные части, начиная от самых нижних частот. Затем, суммируя мощности звука в каждой полосе частот, создаем частотную картину звука.

Этот процесс можно представить на примере нот. Например, кто-то исполняет аккорд До мажор на фортепиано. Этот звук представляет собой комбинацию из трех музыкальных нот – До, Ми и Соль – которые смешивается в один сложный звук. Требуется разбить этот сложный звук на отдельные ноты, чтобы обнаружить исходные. Здесь – то же самое.

Для этого применяется математическая операция, называемая [преобразованием Фурье](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D0%BE%D0%B1%D1%80%D0%B0%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%A4%D1%83%D1%80%D1%8C%D0%B5) [14]. Она раскладывает сложную звуковую волну на простые звуковые волны, которые ее составляют. Имея отдельные звуковые волны, нужно сложить мощности звука в каждой из них. Конечным результатом является оценка важности каждого частотного диапазона, от низких частот до высоких. Для наглядности этих данных, проще всего построить диаграмму, пример такой диаграммы представлен на рисунке 3.

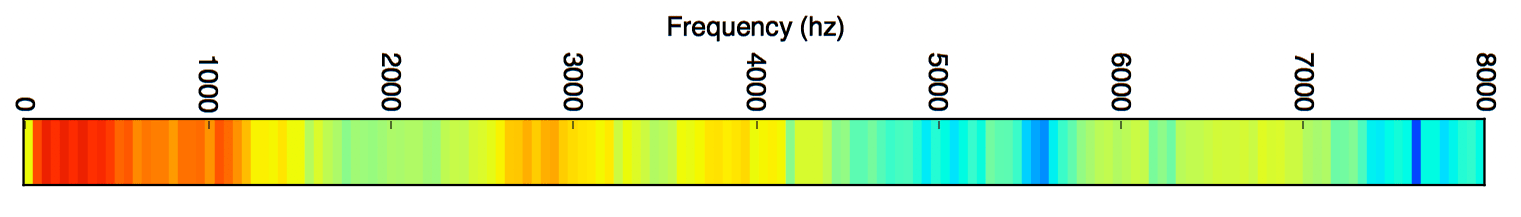


Рисунок 3 – Пример диаграмммы распределения частот

Повторив этот процесс на каждом 20-миллисекундном фрагменте аудио, получим спектрограмму (каждый столбец слева направо представляет собой один 20-миллисекундный фрагмент). С помощью спектрограммы можно выделить музыкальные ноты и другие тона в аудиоданных. Нейронной сети будет проще находить шаблоны в таких данных, чем в сырых записях звука. Пример спектрограммы представлен на рисунке 4.

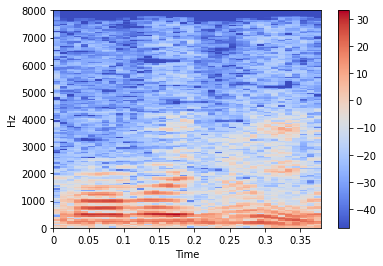


Рисунок 4 – Спектрограмма слова «проводник»

* 1. Сверточная нейронная сеть

Спектрограмму можно воспринимать как изображение звука, поэтому для их классификации возможно использовать сверточные нейронные сети [15]. Они повсеместно используются для решения задач распознавания, после успеха этого типа сетей на ImageNet в 2012 году [16].

В ее основе лежат некоторые особенности [зрительной коры](https://ru.wikipedia.org/wiki/%D0%97%D1%80%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BA%D0%BE%D1%80%D0%B0), в которой были открыты так называемые простые клетки, реагирующие на прямые линии под разными углами, и сложные клетки, реакция которых связана с активацией определенного набора простых клеток. Таким образом, идея сверточных нейронных сетей заключается в чередовании сверточных слоев и субдискретизирующих слоев (слоев подвыборки).

**Сверточный слой**

В сверточном слое используется ограниченная матрица весов небольшого размера, которую «двигают» по всему обрабатываемому слою, формируя после каждого сдвига сигнал активации для нейрона следующего слоя с аналогичной позицией. То есть для различных нейронов выходного слоя используются одна и та же матрица весов, которую также называют ядром свертки. Особенностью сверточного слоя является сравнительно небольшое количество параметров, устанавливаемое при обучении. Так, например, если исходное изображение имеет размерность 100×100 пикселей по трем каналам (это значит 30000 входных нейронов), а сверточный слой использует фильтры c ядром 3x3 пикселя с выходом на 6 каналов, тогда в процессе обучения определяется только 9 весов ядра, однако по всем сочетаниям каналов, то есть 9×3×6 =162, в таком случае данный слой требует нахождения только 162 параметров, что существенно меньше количества искомых параметров полносвязной нейронной сети. Один из этапов процесса свертки представлен на рисунке 5.

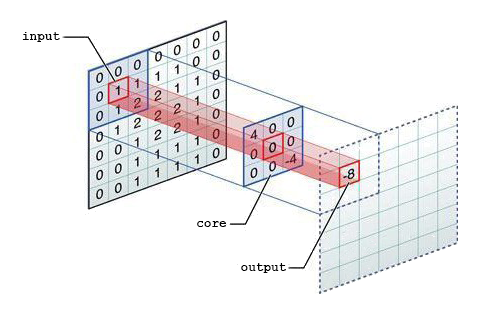


Рисунок 5 – Пример обработки входного слоя ядром свертки

**Слой субдискретизации**

Слой пулинга (иначе подвыборки, субдискретизации) представляет собой нелинейное уплотнение карты признаков, при этом группа пикселей (обычно размера 2×2) уплотняется до одного пикселя, проходя нелинейное преобразование. Наиболее употребительна при этом функция максимума. Преобразования затрагивают непересекающиеся прямоугольники или квадраты, каждый из которых ужимается в один пиксель, при этом выбирается пиксель, имеющий максимальное значение. Операция пулинга позволяет существенно уменьшить пространственный объем изображения. Пулинг интерпретируется так: если на предыдущей операции свертки уже были выявлены некоторые признаки, то для дальнейшей обработки настолько подробное изображение уже не нужно, и оно уплотняется до менее подробного. К тому же фильтрация уже ненужных деталей помогает не [переобучаться](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5). Слой пулинга, как правило, вставляется после слоя свертки перед слоем следующей свертки. Наглядно преобразование можно увидеть на рисунке 6.

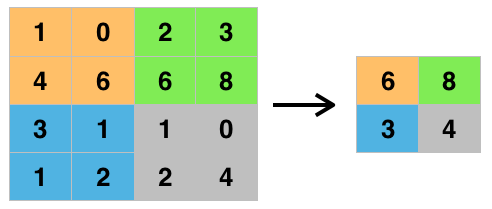


Рисунок 6 – Процесс подвыборки функцией максимума и фильтром 2×2  
с шагом 2

**Выводы по второй главе**

В данной главе был разобран способ представления звуковой информации в компьютере, а также был представлен процесс предварительной ее обработки, для дальнейшего обучения нейросети. Использование спектрограмм аудио файлов должно облегчить обучение нейросети и увеличить точность распознавания. Кроме того, были рассмотрены структура и основные свойства сверточный нейросетей.

1. Проектирование
   1. Функциональные требования

Функциональные требования к разрабатываемому приложению описывают то, как она должна себя вести. Система после разработки должна удовлетворять следующим функциональным требованиям:

1. должен быть реализован графический интерфейс;
2. система должна иметь кнопки запуска/остановки;
3. система должна по голосовой команде запускать стандартные функции Windows:
   * блокнот;
   * проводник;
   * диспетчер задач;
   * настройки;
4. должна быть возможность указания пути к произвольным программам и их запуск по номеру;
5. система должна по голосовой команде сворачивать активное окно;
6. система должна по голосовой команде закрывать активное окно.
   1. Нефункциональные требования

Нефункциональные требования включают в себя те свойства, которые имеются у приложения. Они определяют критерии, которые могут использоваться для оценки работы системы, а не для конкретного поведения. Список нефункциональных требований представлен ниже:

1. система должна быть реализована на языке Python 3.7 [17];
2. система должна использовать библиотеку PyTorch [9] для работы с нейросетями;
3. система должна использовать библиотеку PyQt5 [18] для реализации графического интерфейса;
   1. Диаграмма вариантов использования

Для проектирования системы был использован UML (united modeling language) [19] – универсальный язык моделирования для разработки в области программного обеспечения, предназначенным для обеспечения стандартного способа визуализации проектирования системы. Была построена модель взаимодействия актера с приложением распознавания лица. Диаграмма вариантов использования представлена на рисунке 7.

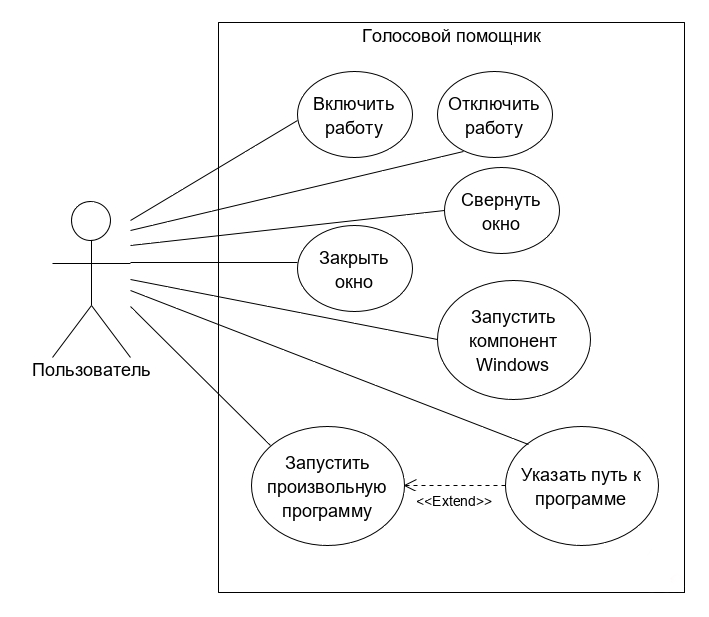


Рисунок 7 – Диаграмма вариантов использования

Главный актер – пользователь, использующий функционал системы. Действия, доступные пользователю описаны ниже.

* + 1. Включить работу. Пользователь может нажать на кнопку запуск, после чего система начнет обрабатывать аудио данные, поступающие с микрофона, и станет доступно использование голосовых команд.
    2. Отключить работу. Пользователь может нажать на кнопку стоп, после чего система прекратит обрабатывать аудио данные, поступающие с микрофона, и использование голосовых команд станет не доступно.
    3. Свернуть окно. Если система запущена, пользователь может отдать голосовую команду «свернуть», после чего система свернет активное окно в панель задач.
    4. Закрыть окно. Если система запущена, пользователь может отдать голосовую команду «закрыть», после чего система закроет активное окно.
    5. Запустить компонент Windows. Если система запущена, пользователь может отдать голосовые команды: «открыть блокнот», «открыть проводник», «открыть диспетчер задач», «открыть настройки». После чего система запустит соответствующий компонент Windows.
    6. Запустить произвольную программу. Если система запущена, пользователь может отдать голосовые команды: «открыть номер один», «открыть номер два», «открыть номер три». После чего система запустит соответствующие программы, если для них был задан путь к исполняющему файлу.
    7. Указать путь к программе. Пользователь может нажать на кнопку обзор и указать путь к исполняющему файлу произвольной программы, после чего система сохранит этот путь для дальнейшего использования.
  1. Топология нейронной сети

Для решения поставленной задачи была разработана нейронная сеть, основа которой была взята из [20]. Она состоит из четырех сверточных слоев, после которых используется функция глобального пулинга и еще два полносвязных слоя. Схема нейронной сети представлена на рисунке 8.

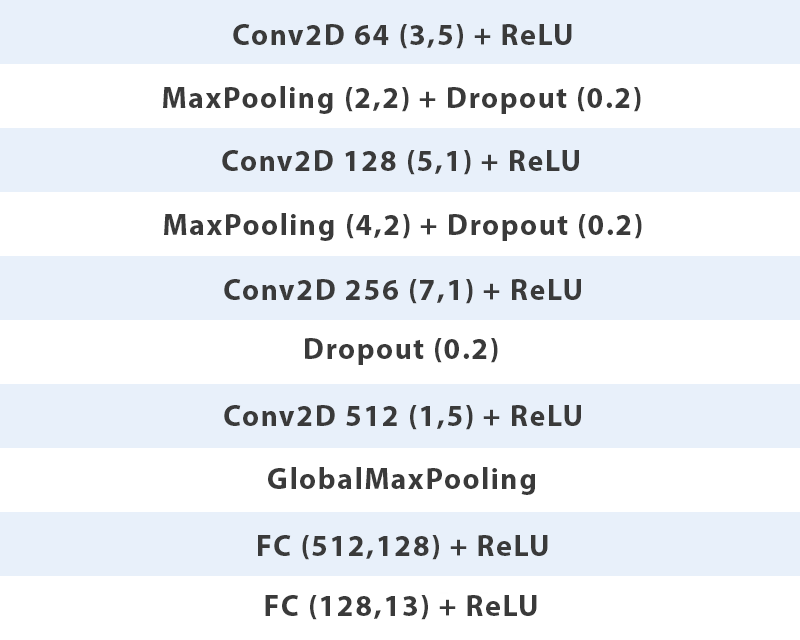


Рисунок 8 – Схема топологии нейронной сети

* 1. Разработка модуля выполнения команд

После обработки данных нейросетью получается число соответствующие определенному слову, эти значения указаны в таблице 1.

Таблица 1 – Соответствие слов числам, полученным от нейросети

|  |  |
| --- | --- |
| **Число** | **Значение** |
| 0 | неопределенное слово |
| 1 | закрыть |
| 2 | свернуть |
| 3 | открыть |
| 4 | номер |
| 5 | один |
| 6 | два |
| 7 | три |
| Окончание таблицы 1 | |
| **Число** | **Значение** |
| 8 | проводник |
| 9 | блокнот |
| 10 | настройки |
| 11 | диспетчер |
| 12 | задач |

Для корректной работы, у модуля есть несколько состояний. Номера и значения состояний приведены в таблице 2.

Таблица 2 – Состояния модуля выполнения команд

|  |  |
| --- | --- |
| **Номер** | **Значение** |
| 0 | Ожидание входящего слова, в этом состоянии выполняются команды закрытия и сворачивания окна. |
| 1 | Это состояние включается, когда было произнесено слово «открыть», ожидание дальнейших команд. |
| 2 | Состояние активируется из состояния 1 после произнесения слова «номер». Ожидается поступление числа, для запуска произвольных программ по номеру. |
| 3 | Состояние активируется из состояния 1 после произнесения слова «диспетчер». Ожидается произнесение слова «задач» для запуска диспетчера задач. |

Для облегчения работы модуля, если значение состояния равно 1 или больше используется таблица 3, содержащая либо новые значения для состояний, либо строку, которую необходимо передать на выполнение. Строки соответствуют номеру состояния, а столбцы номеру поступившего слова.

Таблица 3 – Таблица переходов состояний

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** |
| **1** | 2 | 0 | 0 | 0 | explorer | notepad | control | 3 | 0 |
| **2** | 0 | путь1 | путь2 | путь3 | 0 | 0 | 0 | 0 | 0 |
| **3** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | taskmgr.exe |

Где путь1, путь2, путь3 – путь к произвольным исполняемым файлам, который задается пользователем через графический интерфейс.

Если ячейка таблицы переходов состояний содержит не число, то выполняется команда, содержащаяся в ней, а текущие состояние становится равным 0. Если ячейка содержит число, текущее состояние меняется на соответствующие значение.

Данная таблица должна хранится во внешнем файле. При этом появится возможность быстрого ее редактирования. А также это позволит сохранять заданные пользователем пути к исполняющим файлам произвольных программ. Что значительно упростит для пользователя работу с программой.

**Выводы по третьей главе**

В данном разделе были сформулированы функциональные и нефункциональные требования к системе, а также создана диаграмма вариантов использования. Кроме того, была разработана топология нейронной сети и были определены состояния и таблица переходов состояний для модуля выполнения голосовых команд, реализация которых представлена в следующей главе.

1. Реализация
   1. Программные средства реализации

Для реализации программной части был использован язык программирования Python версии 3.7 [17], использовалась среда разработки JetBrains PyCharm Professional 2020.3.5 [21]. Для обучения нейросети использовалась виртуальная среда Google Colaboratory [22].

Для языка Python были использованы следующие библиотеки:  
PyAudio 0.2.11 [23], NumPy 1.20.2 [24], PyQt5 5.15.4 [18], Librosa 0.8.0 [11], PyTorch 1.8.1 [9].

Для подготовки аудио файлов для обучения использовалась программа обработки аудио Adobe Audition CC 2018 [25]. Также для расширения выборки была использована утилита SoX 14.4.2 [26], позволяющая совершать множество различных операций над аудио файлами.

* 1. Формирование обучающей выборки

Для обучения нейросети аудио данные были взяты из Russian Open Speech To Text (STT/ASR) Dataset [2] и **Russian Open Speech To Text (STT/ASR) Dataset [27]. Они содержат аудиозаписи длиной в несколько секунд и текстовые файлы с содержимым аудиозаписи.**

**Далее с помощью программы Adobe Audition было вручную вырезано около 100 примеров произношения для каждого 12 необходимых слов: «закрыть», «свернуть», «открыть», «номер», «один», «два», «три», «проводник», «блокнот», «настройки», «диспетчер», «задач». А также еще 200 примеров содержащих произношение различных случайных слов.**

**Расширение обучающей выборки**

Для более точного обучения нейросети было принято решение расширить исходную выборку. Для этого использовалась утилита SoX [26] – это свободный кроссплатформенный аудиоредактор, не имеющий графического интерфейса и управляющийся через командную строку, что позволяет использовать его для пакетной обработки через Python. С ее помощью из каждой аудиозаписи было получено еще 3 примера с измененными скоростью и тоном. Скорость была величина или уменьшена на случайную величину от 10% до 25%. Тон был увеличен или уменьшен на случайное значение от 2 до 5 полутонов. Соответствующий исходный код приведен на рисунке 9.

def augment(in\_path, out\_path, dir\_cnt, num):  
   
 for i in range(dir\_cnt):  
 cur\_in\_dir = in\_path + '\\' '{:02}'.format(i)  
 cur\_out\_dir = out\_path + '\\' '{:02}'.format(i)  
 audios = os.listdir(cur\_in\_dir)  
 for audio in audios:  
 cur\_audio\_path = cur\_in\_dir + **'**\\**'** + audio  
 shutil.copy(cur\_audio\_path, cur\_out\_dir + **'**\\**'** + audio)  
 for j in range(num):  
 out\_audio\_path = cur\_out\_dir + **'**\\**'** + str(j) + audio  
 augment\_logic(cur\_audio\_path, out\_audio\_path)  
  
def augment\_logic(in\_str, out\_str):  
  
 if random.randint(0, 1) == 0:  
 stretch\_rate = (random.random() / 4.5) + 1.11  
 else:  
 stretch\_rate = random.randint(750, 900) / 1000  
 if random.randint(0, 1) == 0:  
 pitch\_rate = random.randint(200, 500)  
 else:  
 pitch\_rate = random.randint(-500, -200)  
  
 subprocess.call(["sox", in\_str, out\_str,

"tempo", "-s", '{0:.2f}'.format(stretch\_rate),

"pitch", str(pitch\_rate)])

Рисунок 9 – Исходный код, производящий расширение выборки

**Извлечение признаков**

Как уже говорилось во второй главе, для облегчения обучения нейросети, будут использоваться спектрограммы аудио файлов. Для начала, каждая аудиозапись будет приведена к одинаковой длине, равной 1с. Так как исходные аудиозаписи имеют частоту, равную 16 кГц, то массив чисел, интерпретирующий аудиозапись будет состоять из 16000 цифр. Исходная запись либо дополняется нулями с начала и с конца, если она меньше требуемой длинны, или так же, с концов, обрезается. Для построения спектрограммы использовалась библиотека Librosa [11]. Длина одного фрагмента спектрограммы была установлена равной 20 мс или 320 числам исходного массива, с перекрытием 10 мс. Функция получения спектрограммы приведена на рисунке 10.

def get\_spec(filename, rate):

    x , sr = librosa.load(filename, sr=rate)

    data = np.zeros(sr)

    if x.shape[0] <= sr:

        data[sr // 2 - x.shape[0] // 2 : sr // 2 –

x.shape[0] // 2 + x.shape[0]] = x

    else:

        data = x[(x.shape[0] - sr) // 2 : (x.shape[0] - sr) // 2 + sr]

    X = np.abs(librosa.stft(data, n\_fft=320, hop\_length=160))

    Xdb = librosa.amplitude\_to\_db(X)

    return Xdb

Рисунок 10 – Функция получения спектрограммы аудио файла

* 1. Реализация нейронной сети

Нейросеть была реализована с помощью библиотеки PyTorch [9]. Библиотека позволяет производить вычисления на графических процессорах, что значительно ускоряет обучение, поэтому построение и последующее обучение проводилось на облачной вычислительной платформе Google Collaboratory. Эта платформа предоставляет свои мощности на ограниченное время, для исследовательских целей, в том числе она позволяет использовать и графические процессоры. Исходный код соответствующего класса, реализующего нейросеть, приведен в приложении A.

После расширения обучаемой выборки ее объем составил 5596 примеров аудиозаписей, из них 4197 били использованы для обучения. В качестве функции потерь была использована функция кросс-энтропии, что является стандартным для задач классификации. В качестве оптимизатора использовался Adam, как наиболее универсальный. В параметрах оптимизатора скорость обучения была изменена на 0.5e-5. Обучение проводилось в течении 2000 эпох, в процессе чего веса нейросети с лучшими значениями точности и функции потерь сохранялись. Фрагмент кода, реализующего процесс обучения нейросети представлен на рисунке 11.

for epoch in range(2000):

    print('Epoch', epoch + 1)

    order = np.random.permutation(len(x\_train))

    for start\_index in range(0, len(x\_train), batch\_size):

        optimizer.zero\_grad()

        batch\_indexes = order[start\_index:start\_index+batch\_size]

        X\_batch = x\_train[batch\_indexes].to(device)

        y\_batch = y\_train[batch\_indexes].to(device)

        preds = mynet2.forward(X\_batch)

        loss\_value = loss(preds, y\_batch)

        loss\_value.backward()

        optimizer.step()

    cnt = 0

    sum\_batch\_loss = 0

    sum\_batch\_acc = 0

    for start\_index in range(0, len(x\_test), batch\_size):

        cnt += 1

        X\_batch = x\_test[start\_index:min(start\_index+batch\_size,

len(x\_test))].to(device)

        y\_batch = y\_test[start\_index:min(start\_index+batch\_size,

len(x\_test))].to(device)

        test\_preds = mynet2.forward(X\_batch)

        sum\_batch\_loss += loss(test\_preds, y\_batch).data.cpu()

        sum\_batch\_acc += (test\_preds.argmax(dim=1) == y\_batch).float().mean().data.cpu()

    cur\_loss = sum\_batch\_loss / cnt

    if accuracy >= 0.8:

        save\_name = '/content/drive/MyDrive/Dataset/newWeights3/acc=' +

str(float(accuracy)) + '\_loss=' +

str(float(cur\_loss)) + '\_ep=' + str(epoch) + '.pth'

        if accuracy > max\_acc:

            torch.save(mynet2.state\_dict(), save\_name)

            max\_acc = accuracy

        elif cur\_loss < min\_loss:

            torch.save(mynet2.state\_dict(), save\_name)

min\_loss = cur\_loss

    print('Accuracy = ' + str(accuracy) + '\tLoss = ' + str(cur\_loss))

Рисунок 11 – Фрагмент кода, реализующего обучение нейросети

* 1. Реализация распознавания слов

Для реализации модуля распознавания слов была использована библиотека PyAudio [23], она позволяет обрабатывать поток аудио данных, передающихся с записывающих устройств. Программный модуль отслуживает уровень громкости звука, поступающего с записывающих устройств, и, если он превышает заданный параметр, сохраняет данные в массив. Если уровень громкости находится ниже заданного параметра в течении 0,2 с сохранение данных прекращается. Если сохраненный фрагмент по времени длится меньше 0,2 с или больше 1 с, то он игнорируется. Если фрагмент подходит по времени в нему добавляется по 0,2 с данных записаны до и после этого фрагмента, получившийся массив сохраняется для дальнейшей обработки. Исходный код класса, реализующего данный программный модуль приведен в приложении Б.

Далее полученные данные обрабатываются нейросетью, для этого также строится спектрограмма полученного аудио фрагмента. На выходе из нейросети получается вектор из 13 чисел. Индекс максимального числа будет соответствовать наиболее вероятному слову. Исходный код, реализующий эту функцию представлен на рисунке 12.

def predict(self, x):  
  
 data = np.zeros(rate)  
 if x.shape[0] <= rate:  
 data[rate // 2 - x.shape[0] // 2 : rate // 2 - x.shape[0] // 2 +

x.shape[0]] = x  
 else:  
 data = x[(x.shape[0] - rate) // 2 : (x.shape[0] - rate) // 2 +

rate]  
  
 X = np.abs(librosa.stft(data, n\_fft=321, hop\_length=160))  
 Xdb = librosa.amplitude\_to\_db(X)[:160]  
  
 t\_tens = torch.FloatTensor(Xdb)  
 t\_tens = t\_tens.unsqueeze(0).float()  
 t\_tens = t\_tens.unsqueeze(0).float()  
 preds = self.net.forward(t\_tens)  
  
 return int(preds[0].argmax().data)

Рисунок 12 – Функция распознавания слов нейросетью

* 1. Реализация модуля выполнения команд

Для реализации модуля выполнения команд использовалась библиотека pywin32 [28], Обеспечивающая доступ к большей части API Win32 и дающая возможность создавать и использовать COM-объекты. А также модуль system встроенной библиотеки os, позволяющий выполнять консольные команды. Соответствующий фрагмент кода приведен на рисунке 13.

while not self.stopped():  
 audio\_lock.acquire()  
 n = len(audios)  
 audio\_lock.release()  
 if n > 0:  
 audio\_lock.acquire()  
 x = audios[0].copy()  
 del audios[0]  
 audio\_lock.release()  
  
 word = self.predict(x)

if st == 0 and word == 1:  
 win32gui.PostMessage(win32gui.GetForegroundWindow(),

win32con.WM\_CLOSE, 0, 0)  
 elif st == 0 and word == 2:  
 win32gui.ShowWindow(win32gui.GetForegroundWindow(),

win32con.SW\_MINIMIZE)  
 elif st == 0 and word == 3:  
 st = 1  
 elif st >= 1 and word >= 4 and word <= 12:  
 command = self.matrix[st - 1][word - 4]  
 if command.isdigit():  
 st = int(command)  
 else:  
 os.system(command)  
 st = 0  
 else:  
 st = 0

Рисунок 13 – Исходный код, отвечающий за выполнение команд

В данном фрагменте переменная st хранит в себе номер состояния, а word – номер слова. Переменная matrix хранит в себе таблицу переходов состояний, описанную в третьей главе, которая подгружается из json файла. Пример содержимого файла представлен на рисунке 14.

[

["2", "0", "0", "0", "explorer", "notepad", "control", "3", "0"],  
 ["0", "D:/SUSU/4 year/BIS/Sem2/Lab1/dist/main.exe", "\"C:/Program Files (x86)/Microsoft Office/root/Office16/WINWORD.EXE\"", "\"E:/Program Files (x86)/WinDjView/WinDjView.exe\"", "0", "0", "0", "0", "0"],  
 ["0", "0", "0", "0", "0", "0", "0", "0", "taskmgr.exe"]

]

Рисунок 14 – Содержимое json файла, хранящего таблицу переходов состояний

* 1. Реализация графического интерфейса

Для реализации графического интерфейса программы использовался Qt Designer [29] – среда разработки графического интерфейса для программ, использующих библиотеку Qt [30], в том числе PyQt5 [18] для Python.

Интерфейс содержит кнопки запуска и остановки исполнения голосовых команд, а также кнопки для указания пути к исполняемым файлам произвольных программ. Скриншот графического интерфейса изображен на рисунке 15.

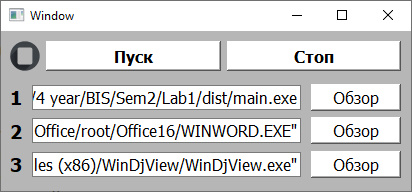


Рисунок 15 – Скриншот графического интерфейса

**Выводы по четвертой главе**

В данной главе были описаны используемые средства разработки, процесс создания обучающей выборки и реализация нейронной сети. Также приведен программный код для модулей распознавания слов и выполнения команд. Кроме того, показан графический интерфейс приложения. Разработанное приложение полностью соответствует ранее упомянутым требованиям.

1. Тестирование
   1. Тестирование нейронной сети

Для тестирования было использовано 20% от общей выборки изображений, не участвовавших в обучении, всего 1399 примеров. Точность нейросети при минимальном значении функции потерь составила 94,07%. График точности представлен на рисунке 16.

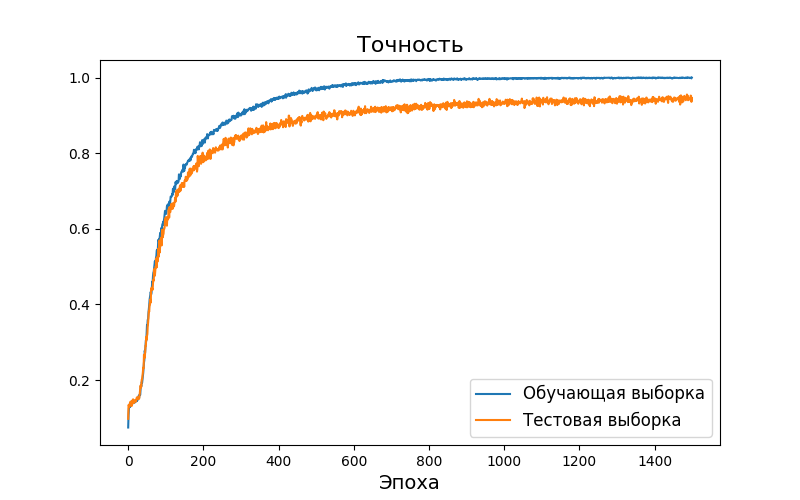


Рисунок 16 – График точности нейросети

* 1. Функциональное тестирование

Было проведено функциональное тестирование системы согласно описанным ранее требованиям. Результаты тестирования приведены в таблице 4.

Таблица 4 – Результаты функционального тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Действие** | **Ожидаемый результат** | **Результат теста** |
| 1 | Включить работу | После нажатая кнопки «Пуск», программа начинает обрабатывать поступающие голосовые команды. | Пройден |
| 2 | Отключить работу | После нажатия кнопки «Стоп», программа перестает обрабатывать поступающие голосовые команды. | Пройден |
| Окончание таблицы 4 | | | |
| № | **Действие** | **Ожидаемый результат** | **Результат теста** |
| 3 | Свернуть окно | После голосовой команды «свернуть», программа сворачивает активное окно в панель задач. | Точность выполнения команды составила 85%. |
| 4 | Закрыть окно | После голосовой команды «закрыть», программа закрывает активное окно. | Точность выполнения команды составила 90% |
| 5 | Запустить блокнот | После голосовой команды «открыть блокнот», программа запускает блокнот. | Точность выполнения команды составила 90% |
| 6 | Запустить диспетчер задач | После голосовой команды «открыть диспетчер задач», программа запускает диспетчер задач. | Точность выполнения команды составила 70%. |
| 7 | Запустить настройки | После голосовой команды «отрыть настройки», программа запускает настройки Windows. | Точность выполнения команды составила 70% |
| 8 | Запустить проводник | После голосовой команды «отрыть проводник», программа запускает проводник Windows. | Точность выполнения команды составила 85% |
| 9 | Запустить произвольную программу | После голосовых команд «отрыть номер один», «отрыть номер два», отрыть номер три», программа запускает произвольную программа, если был задан путь к исполняемому файлу. | Точность выполнения команды составила 60% |
| 10 | Указать путь к исполняемому файлу произвольной программы | После нажатия кнопки «Обзор», открывается окно выбора файлов, в котором можно выбрать исполняемый файл. После повторного запуска программы заданный путь должен сохраняться. | Пройден |

**Выводы по пятой главе**

В данной главе приведены результаты тестирования нейронной сети, являющейся основой программы. Также было проведено функционального тестирования основной программы, в целом результаты удовлетворительные.

Заключение

В рамках данной работы была спроектирована и реализована программа голосового помощника для ОС Windows, позволяющего упростить использование ее простейших функций. для распознания жильцов, входящих в подъезд. В основе программы лежит использование нейронных сетей для распознавания голосовых команд. Также было проведено тестирование нейронной сети и функциональное тестирование системы.

В ходе работы были решены следующие задачи:

1. проведен обзор аналогов и научной литературы;
2. подготовлена обучающая и тестовая выборка аудио файлов;
3. программно реализована и обучена нейронная сеть;
4. разработано и протестировано приложение.

Также были улучшены навыки программирования на языке Python, получен опыт работы с нейронными сетями с помощью библиотеки PyTorch, получены навыки записи и обработки звука с помощью библиотек Librosa и PyAudio.

литература

* + 1. Хайкин С. Нейронные сети: полный курс. 2-е изд. Москва: ООО "И.Д. Вильямс", 2017 г. 1104 с.
    2. Russian Open Speech To Text (STT/ASR) Dataset [Электронный ресурс] URL: https://github.com/snakers4/open\_stt/ (дата обращения: 28.03.2021).
    3. Егунов, В. А. Распознавание речевых команд с использованием нейронных сетей / В. А. Егунов, С. В. Панюлайтис // Прикаспийский журнал: управление и высокие технологии. – 2020 г. – № 3(51). – С. 53-61.
    4. Кипяткова, И. С. Разновидности глубоких искусственных нейронных сетей для систем распознавания речи / И. С. Кипяткова, А. А. Карпов // Труды СПИИРАН. – 2016 г. – № 6(49). – С. 80-103. – DOI 10.15622/sp.49.5.
    5. Гапочкин, А. В. Нейронные сети в системах распознавания речи / А. В. Гапочкин // Science Time. – 2014 г. – № 1. – С. 29-36.
    6. Cortana – Your personal productivity assistant [Электронный ресурс] URL: https://www.microsoft.com/en-us/cortana/ (дата обращения: 18.05.2021).
    7. Yandex SpeechKit | Yandex.Cloud – Документация [Электронный ресурс] URL: https://cloud.yandex.ru/docs/speechkit/ (дата обращения: 18.05.2021).
    8. Speereo Voice Assistant [Электронный ресурс] URL: http://www.speereo.com/ (дата обращения: 18.05.2021).
    9. Документация PyTorch [Электронный ресурс] URL: https://pytorch.org/docs/stable/index.html (дата обращения: 26.04.2021).
    10. Официальный сайт пакета Scikit-learn [Электронный ресурс] URL: https://scikit-learn.org/stable/ (дата обращения: 18.05.2021).
    11. Документация Librosa [Электронный ресурс] URL: https://librosa.org/doc/latest/index.html (дата обращения: 30.03.2021)
    12. Документация SciPy [Электронный ресурс] URL: https://docs.scipy.org/doc/scipy/reference/index.html (дата обращения: 30.03.2021)
    13. Представление звуковой информации в компьютере [Электронный ресурс] URL: https://www.yaklass.ru/p/informatika/10-klass/informatciia-i-informatcionnye-protcessy-11955/predstavlenie-nechislovoi-informatcii-v-kompiutere-12433/re-b2f76748-2c67-4268-8bc3-c1227079274c/ (дата обращения: 19.05.2021)
    14. Брейсуэлл Р. Н. Преобразование Фурье // Scientific American. В мире науки. – 1989 г. – № 8. – С. 48–56.
    15. LeCun Y., Bengio Y. Convolution Networks for Images, Speech, and Time-Series // Igarss 2014 г, No. 1, 1998. pp. 1-5.
    16. Large Scale Visual Recognition Challenge [Электронный ресурс] URL: http://image-net.org/challenges/LSVRC/2012/results.html#t1 (дата обращения: 11.09.2020).
    17. Welcome to Python.org [Электронный ресурс] URL: https://www.python.org/ (дата обращения: 21.05.2021)
    18. PyQt5 ⋅ PyPl [Электронный ресурс] URL: https://pypi.org/project/PyQt5/ (дата обращения: 21.05.2021)
    19. Официальный сайт UML [Электронный ресурс] URL: https://www.uml.org (дата обращения: 11.09.2020).
    20. Как использовать распознавание речи нейросетью в чат-ботах [Электронный ресурс] URL: https://www.azoft.ru/blog/how-to-train-a-neural-network-to-recognize-speech/ (дата обращения: 22.03.2021).
    21. PyCharm: IDE для профессиональной разработки на Python от JetBrains [Электронный ресурс] URL: https://www.jetbrains.com/ru-ru/pycharm/ (дата обращения: 22.05.2021).
    22. Google Colaboratory [Электронный ресурс] URL: https://colab.research.google.com/notebooks/intro.ipynb?utm\_source=scs-index (дата обращения: 22.03.2021).
    23. PyAudio Documentation – PyAudio 0.2.11 documentation [Электронный ресурс] URL: https://people.csail.mit.edu/hubert/pyaudio/docs/ (дата обращения: 22.05.2021).
    24. NumPy [Электронный ресурс] URL: https://numpy.org/ (дата обращения: 22.05.2021).
    25. Программа для аудиозаписи и аудиомонтажа | Adobe Audition [Электронный ресурс] URL: https://www.adobe.com/ru/products/audition.html (дата обращения: 22.05.2021)
    26. SoX - Sound eXchange | HomePage [Электронный ресурс] URL: http://sox.sourceforge.net/ (дата обращения: 07.05.2021)
    27. Russian Open Speech To Text (STT/ASR) Dataset | Kaggle [Электронный ресурс] URL: https://www.kaggle.com/tapakah68/audio-dataset (дата обращения: 20.04.2021)
    28. Python for Win32 Extensions Help [Электронный ресурс] URL: http://timgolden.me.uk/pywin32-docs/PyWin32.html (дата обращения: 20.04.2021)
    29. Qt Designer Manual [Электронный ресурс] URL: https://doc.qt.io/qt-5/qtdesigner-manual.html (дата обращения: 22.05.2021)
    30. Qt | Cross-platform software development for embedded & desktop [Электронный ресурс] URL: https://www.qt.io/ (дата обращения: 22.05.2021)

Приложение

Приложение А. Код класса нейросети

Листинг 1 – Класс для описания нейросети

class MySTTnet(nn.Module):  
 def \_\_init\_\_(self):  
 super(MySTTnet, self).\_\_init\_\_()  
  
 self.conv1 = nn.Conv2d(in\_channels=1, out\_channels=64, kernel\_size=(3,5), padding=(1,0))  
 self.act1 = nn.ReLU()  
 self.pool1 = nn.MaxPool2d(kernel\_size=(2,2), stride=(2,2))  
 self.drop1 = nn.Dropout2d(p=0.2)  
  
 self.conv2 = nn.Conv2d(in\_channels=64, out\_channels=128, kernel\_size=(5,1), padding=(2,0))   
 self.act2 = nn.ReLU()  
 self.pool2 = nn.MaxPool2d(kernel\_size=(4,2), stride=(4,2))   
 self.drop2 = nn.Dropout2d(p=0.2)  
  
 self.conv3 = nn.Conv2d(in\_channels=128, out\_channels=256, kernel\_size=(7,1), padding=0)  
 self.act3 = nn.ReLU()  
 self.conv4 = nn.Conv2d(in\_channels=256, out\_channels=512, kernel\_size=(1,5), padding=0)  
 self.act4 = nn.ReLU()  
 self.pool3 = nn.MaxPool2d(kernel\_size=(14,20), stride=(14,20))  
 self.drop3 = nn.Dropout2d(p=0.2)  
  
 self.fc1 = nn.Linear(512, 128) # (1728,)  
 self.act6 = nn.ReLU()  
 self.fc2 = nn.Linear(128, 15) # (1728,)  
  
 def forward(self, x):  
  
 x = self.conv1(x)  
 x = self.act1(x)  
 x = self.pool1(x)  
 x = self.drop1(x)  
  
 x = self.conv2(x)  
 x = self.act2(x)  
 x = self.pool2(x)  
 x = self.drop2(x)  
  
 x = self.conv3(x)  
 x = self.act3(x)  
 x = self.conv4(x)  
 x = self.act4(x)  
 x = self.pool3(x)  
 x = self.drop3(x)  
  
 x = x.view(x.size(0), x.size(1) \* x.size(2) \* x.size(3))  
  
 x = self.fc1(x)  
 x = self.act6(x)  
 x = self.fc2(x)  
  
 return x

Приложение Б. Код модуля распознавания слов

Листинг 2 – Модуль распознавания слов

import pyaudio  
import math  
import struct  
import wave  
import time  
import os  
from threading import Thread, Lock, Event  
import numpy as np  
  
Threshold = 12  
  
SHORT\_NORMALIZE = (1.0/32768.0)  
chunk = 1024  
FORMAT = pyaudio.paInt16  
CHANNELS = 1  
RATE = 16000  
swidth = 2  
  
TIMEOUT\_LENGTH = 0.2  
f\_name\_directory = r'D:\SUSU\Diplom\Code\test'  
  
class Recorder(Thread):  
  
 @staticmethod  
 def rms(frame):  
 count = len(frame) / swidth  
 format = "%dh" % (count)  
 shorts = struct.unpack(format, frame)  
  
 sum\_squares = 0.0  
 for sample in shorts:  
 n = sample \* SHORT\_NORMALIZE  
 sum\_squares += n \* n  
 rms = math.pow(sum\_squares / count, 0.5)  
  
 return rms \* 1000  
  
 def \_\_init\_\_(self, audios:list, audio\_lock:Lock):  
 Thread.\_\_init\_\_(self)  
 self.daemon = True  
 self.audios = audios  
 self.audio\_lock = audio\_lock  
 self.p = pyaudio.PyAudio()  
 self.\_stop = Event()  
 self.stream = self.p.open(format=FORMAT,  
 channels=CHANNELS,  
 rate=RATE,  
 input\_device\_index=2,  
 input=True,  
 output=True,  
 frames\_per\_buffer=chunk)  
  
 def stop(self):  
 self.\_stop.set()  
 print('Listening stop')  
  
 def stopped(self):  
 return self.\_stop.isSet()  
  
 def record(self, rec:list):  
 print('Noise detected, recording beginning')  
 #rec = []  
 current = time.time()  
 end = time.time() + TIMEOUT\_LENGTH  
 for\_save = []  
  
 while current <= end:  
  
 data = self.stream.read(chunk)  
 if self.rms(data) >= Threshold:  
 end = time.time() + TIMEOUT\_LENGTH  
  
 fr = np.frombuffer(data, np.int16)  
 rec.extend(list(fr))  
  
 for\_save.append(data)  
 current = time.time()  
  
 rec = (np.array(rec, dtype=float) / (2 \*\* 15))[:len(rec) - int(RATE \* 0.1)]  
 if rec.shape[0] > RATE \* 0.25 and rec.shape[0] < RATE:  
 self.audio\_lock.acquire()  
 self.audios.append(rec)  
 self.audio\_lock.release()  
  
  
 def write(self, recording):  
 n\_files = len(os.listdir('test'))  
  
 filename = os.path.join('test', '{}.wav'.format(n\_files))  
  
 wf = wave.open(filename, 'wb')  
 wf.setnchannels(CHANNELS)  
 wf.setsampwidth(self.p.get\_sample\_size(FORMAT))  
 wf.setframerate(RATE)  
 wf.writeframes(recording)  
 wf.close()  
 print('Written to file: {}'.format(filename))  
 print('Returning to listening')  
  
  
 def run(self):  
 print('Listening beginning')  
 buffer = [[], [], []]  
 while not self.stopped():  
  
 input = self.stream.read(chunk)  
 buffer[0] = buffer[1]  
 buffer[1] = buffer[2]  
 buffer[2] = input  
 rms\_val = self.rms(input)  
 if rms\_val > Threshold:  
 b = list(np.frombuffer(buffer[0], np.int16))  
 b.extend(list(np.frombuffer(buffer[1], np.int16)))  
 b.extend(list(np.frombuffer(buffer[2], np.int16)))  
 self.record(b)