**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc58155733)

[1 ПОСТАНОВКА ЗАДАЧИ 5](#_Toc58155734)

[2 РАЗРАБОТКА АЛГОРИТМОВ 6](#_Toc58155735)

[3 РАЗРАБОТКА ПРОГРАММЫ 9](#_Toc58155736)

[4 ТЕСТИРОВАНИЕ 13](#_Toc58155737)

[4.1 Описание входных и выходных данных 13](#_Toc58155738)

[4.2 Результаты тестирования 14](#_Toc58155739)

[ЗАКЛЮЧЕНИЕ 19](#_Toc58155740)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 20](#_Toc58155741)

ПРИЛОЖЕНИЕ А ТЕКСТ ПРОГРАММЫ

ПРИЛОЖЕНИЕ Б ГРАФИЧЕСКИЙ МАТЕРИАЛ

ВВЕДЕНИЕ

В настоящее время компьютерные системы играют важную роль в нашей жизни. Общение человека с компьютером в большинстве случаев осуществляется посредством использования клавиатуры и мыши. Однако более удобным способом взаимодействия с компьютером является речь. Распознавание речи делает это возможным.

На сегодняшний день исследования распознавания речи не только не потеряли актуальность, но и активно развиваются, расширяя области своего практического применения.

Распознавание речи, более известное как автоматическое распознавание речи (ASR) представляет собой процесс интерпретации человеческой речи в компьютере. Это построение схемы отображения акустических сигналов [2].

Проблема автоматического распознавания речи с помощью компьютера сложна из-за сложности человеческого языка.

Основная трудность, которая препятствует внедрению речевых технологий в различные сферы деятельности и быта человека, это недостаточная устойчивость процесса распознавания и понимания речи. Созданию методов понимания речи, устойчивых к различным видам искажений (внешние шумы, вариации произношения, синтаксические отклонения и др.) посвящена работа большого количества специалистов по всему миру. Было разработано большое количество алгоритмов для распознавания отдельно произносимых слов. Однако раздельное произношение в отличие от слитной речи существенно замедляет и усложняет речевой диалог между пользователем и компьютером.

1. ПОСТАНОВКА ЗАДАЧИ

Разработать приложение (консольное или оконное), позволяющее распознавать голосовые команды на английском языке, записанные в файле с заданным форматом (.wav). Приложение должно использовать предобученную модель распознавания английской речи.

Обучить нейронную сеть для распознавания нескольких (10-20) голосовых команд, используя подготовленную обучающую выборку. При обучении использовать любые доступные базы данных. Предусмотреть возможность работы с файлами, записанными с использованием внешнего микрофона.

Протестировать работу обученной модели для распознавания английской речи.

1. РАЗРАБОТКА АЛГОРИТМОВ

Разрабатываемая нейронная сеть – это end-to-end автоматическое распознавание речи (Automatic Speech Recognition, ASR). Это система, которая предназначена для того, чтобы напрямую отражать последовательность акустических признаков в последовательности графем (букв) или слов.

Завершенный конвейер будет принимать необработанный звук в качестве входного сигнала и возвращать прогнозируемую транскрипцию разговорного языка. Полный конвейер представлен на рисунке 2.1.

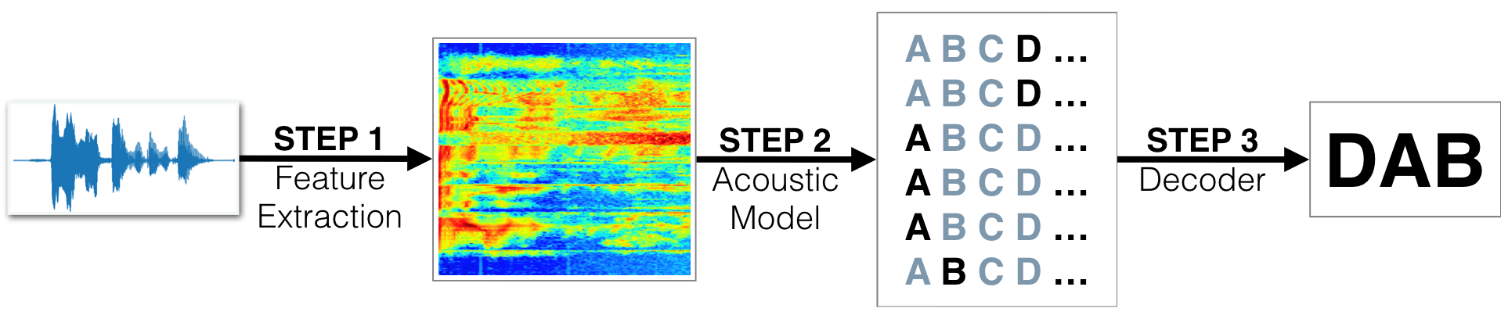


Рисунок 2.1 – Структура end-to-end модели

Шаг 1. Предварительная обработка входных данных. Преобразование необработанного звука в нормализованную спектрограмму.

Шаг 2. Акустическая модель. На вход подаются обработанные звуковые сигналы. На выходе формируется распределение вероятностей по всем потенциальным транскрипциям.

Разрабатываемая модель основана на исследованиях Baidu's «Deep Speech 2: End-to-End Speech Recognition in English and Mandarin» [3], но имеет упрощенную структуру (рис. 2.2).

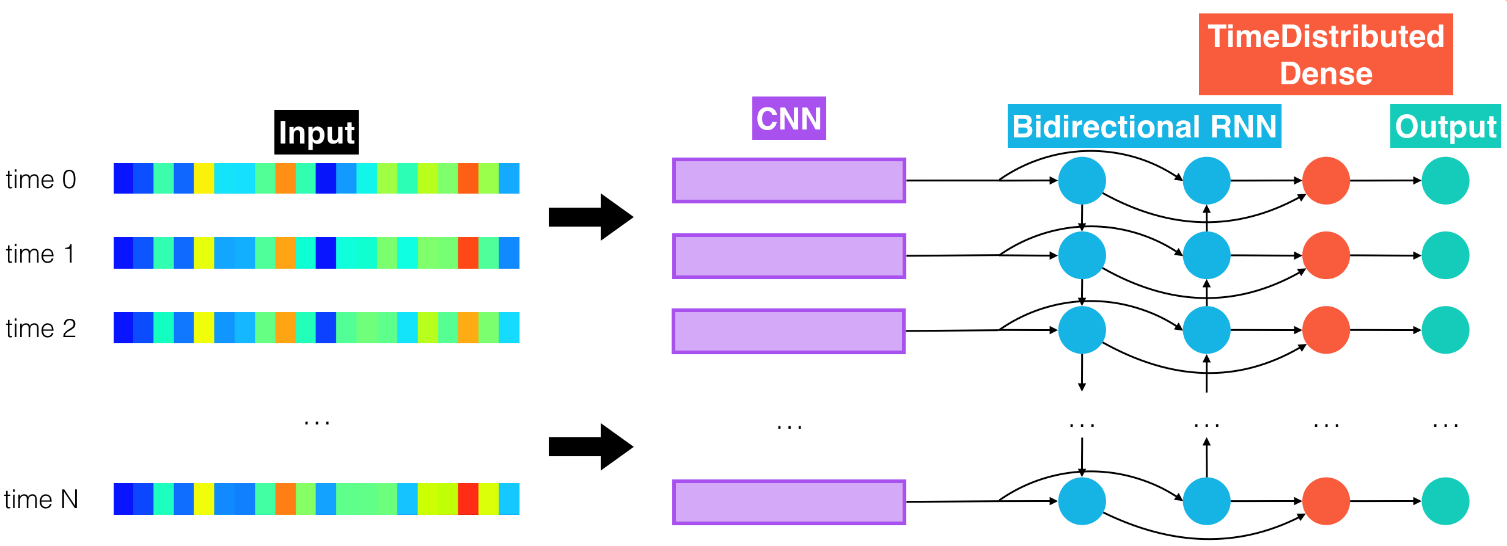


Рисунок 2.2 – Архитектура модели

Структура:

* + - 1. Входные данные (Input) – необработанные звуковые сигналы.
      2. 1D сверточный слой (CNN) преобразует звуковые данные в одномерный тензор.
      3. Пакетная нормализация (Batch normalization) выравнивает дисперсию звуковых сигналов, что также должно сократить время обучения и привести к более быстрой сходимости, требуя меньшего количества слоев, чем ненормализованные данные.
      4. Исключение (Dropout) используется для уменьшения переобучения модели.
      5. 4 слоя двунаправленных рекуррентных нейронных сетей (Bidirectional RNN) с пакетной нормализацией и исключением на крайних слоях. С помощью двунаправленности увеличивается объем информации, доступной для рекуррентной сети за счет связывания текущего состояния с предыдущим и следующим. Наличие данной информации помогает сети делать более точные прогнозы.
      6. Распределенный по времени слой (TimeDistributed Dense), применяемый к конечному слою нормализации. Он применяет слой к каждому временному срезу звуковой волны. Входные данные в модель являются последовательностями. Для достижения более точной модели каждая часть последовательности должна быть рассмотрена, что позволяет сделать распределенный во времени слой.
      7. Слой активации Softmax применяется к модели для классификации каждого из символов. Задача модели состояла в том, чтобы классифицировать каждую из временных звуковых последовательностей на символы. Функция softmax была разработана для того, чтобы хорошо работать при категоризации входных данных. Следовательно, функция softmax была подходящим решением для активационного слоя модели.

Softmax имеет 29 выходных классов: 26 английских букв, пробел, апостроф и пустой символ.

Шаг 3. Формирование прогнозируемой транскрипции разговорного языка. Конвейер получает на вход выходные данные акустической модели и возвращает прогнозируемую транскрипцию.

1. РАЗРАБОТКА ПРОГРАММЫ

Для решения задачи обнаружения объектов в данном проекте будем использовать язык Python, так как он кроссплатформенный – хорошо работает на всех популярных ОС и переносится на них без каких-либо доработок.

Среди большого количества инструментального ПО была выбрана среда разработки JetBrains PyCharm она удобна в работе и имеет бесплатную версию с функционалом, которого вполне достаточно. Кроме интерпретатора python были использованы библиотека глубокого обучения Keras [1].

Для обучения и тестирования также применялся Jupyter-ноутбук, позволяющий хранить вместе код, изображения, комментарии, формулы и графики. С его помощью сформировано наглядное представление проведенной работы.

В таблице 3.1 описаны основные модули программы.

Таблица 3.1 – Описание модулей программы

|  |  |  |
| --- | --- | --- |
| **Название модуля** | **Методы модуля** | **Описание модуля** |
| 1 | 2 | 3 |
| convert\_flac\_to\_wav.py | - | Утилита для преобразования аудиофайлов с расширением .flac в аудиофайлы с расширением .wav. |
| create\_desc\_json.py | - | Утилита для формирования файлов описания обучающей выборки, которые содержат относительный путь к аудиофайлам, их продолжительность и расшифровку. |
| char\_map.py | - | Утилита для формирования словарей графема-код, код-графема для преобразования графем в коды и наоборот. |

*Продолжение таблицы 3.1*

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| predict.py | get\_prediction() | Утилита для распознавания голосовых команд. |
| utils.py | spectrogram()  spectrogram\_from\_file()  text\_to\_int\_sequence()  int\_sequence\_to\_text() | Функции для обработки входных и выходных данных. |
| plot\_utils.py | plot\_train\_valid\_loss()  plot\_raw\_audio()  plot\_spectrogram\_feature() | Функции для построения графиков. |
| train\_utils.py | ctc\_lambda\_func()  add\_ctc\_loss()  train\_model() | Утилита для обучения нейронной сети. |
| sample\_model.py | cnn\_output\_length()  final\_model() | Описание модели нейронной сети. |
| data\_generator.py | DataGenerator() get\_batch() shuffle\_data\_by\_partition() next\_train() next\_valid() load\_train\_data() load\_validation\_data() load\_metadata\_from\_desc\_file() featurize() normalize() | Подготавливает данные для обучения нейронной сети. |

Описание функций модуля predict.py, принимаемых аргументов и возвращаемых значений в таблице 3.2.

Таблица 3.6 – Аргументы модуля predict.py

|  |  |
| --- | --- |
| **Аргументы** | **Пояснение** |
| get\_prediction | Формирует проверочную тран-скрипцию разговорного языка |

Описание функций модуля train\_utils.py, принимаемых аргументов и возвращаемых значений в таблице 3.5.

Таблица 3.5 – Аргументы модуля train\_utils.py

|  |  |
| --- | --- |
| **Аргументы** | **Пояснение** |
| ctc\_lambda\_func | Функция для вычисления ctc потерь |
| add\_ctc\_loss | Функция для добавления ctc\_lambda\_func в модель |
| train\_model | Функция для обучения модели |

Описание функций модуля utils.py, принимаемых аргументов и возвращаемых значений в таблице 3.3.

Таблица 3.3 – Аргументы модуля utils.py

|  |  |
| --- | --- |
| **Аргументы** | **Пояснение** |
| spectrogram | Формирует нормализованную мел-спектрограмму |
| spectrogram\_from\_file | Формирует мел-спектрограмму из файла |
| text\_to\_int\_sequence | Преобразует текстовую строку в числовую последовательность |
| int\_sequence\_to\_text | Преобразует числовую последователь-ность в текстовую строку |

Описание функций модуля plot\_utils.py, принимаемых аргументов и возвращаемых значений в таблице 3.4.

Таблица 3.4 – Аргументы модуля plot\_utils.py

|  |  |
| --- | --- |
| **Аргументы** | **Пояснение** |
| plot\_train\_valid\_loss | Формирует график функции потерь для обучения и прогнозирования |
| plot\_raw\_audio | Формирует график отображения необработанного звука |
| plot\_spectrogram\_feature | Формирует график нормализованной мел-спектрограммы |

Описание функций модуля sample\_model.py, принимаемых аргументов и возвращаемых значений в таблице 3.6.

Таблица 3.6 – Аргументы модуля sample\_model.py

|  |  |
| --- | --- |
| **Аргументы** | **Пояснение** |
| final\_model | Формирует модель нейронной сети для распознавания речи |
| cnn\_output\_length | Формирует длину выходной после-довательности 1D сверточного слоя |

Описание функций модуля data\_generator.py, принимаемых аргументов и возвращаемых значений в таблице 3.7.

Таблица 3.7 – Аргументы модуля data\_generator.py

|  |  |
| --- | --- |
| **Аргументы** | **Пояснение** |
| get\_batch | Получение пакета данных для обучающей, проверочной и тес-товой выборки |
| shuffle\_data\_by\_partition | Функция для перемешивания данных |
| next\_train | Получение пакета данных для обучающей выборки |
| next\_valid | Получение пакета данных для проверочной выборки |
| load\_train\_data | Загрузка данных из описательных файлов для тренировочного набора |
| load\_validation\_data | Загрузка данных из описательных файлов для проверочного набора |
| load\_metadata\_from\_desc\_file | Загрузка данных из описательных файлов (относительный путь к аудио-файлам, их продолжительность и расшифровку) |
| featurize | Функция для преобразования аудио-файла в мел-спектрограмму |
| normalize | Функция для нормализации мел-спектрограммы |

1. ТЕСТИРОВАНИЕ
   1. Описание входных и выходных данных

В качестве обучающей выборки используется набор данных Librispeech [4], который в общей сложности содержит тысячу часов английской речи с частотой 16 кГц, собранных из аудиокниг. Обучение и проверка проводились на выборках по 5.4 часа каждая. Для обучающей выборки были сформированы файлы описания, которые содержат относительный путь к аудиофайлам, их продолжительность и расшифровку (рис 4.1).

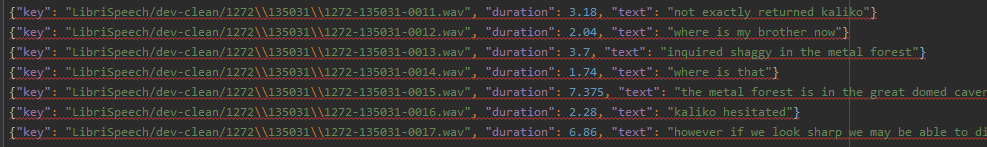


Рисунок 4.1 – Часть файла описания для обучающей выборки

Входными данными являются подготовленные аудиофайлы, представленные в виде нормализованной спектрограммы. Выходными данными – прогнозируемые транскрипции разговорного языка.

* 1. Результаты тестирования

Среда тестирования – ПК, процессор Intel Core i5-8250U с частотой 1.6 ГГц, ОЗУ 8 ГБ, тип системы: 64-разрядная OC Windows10.

Обучение и тестирование производилось на базе Jupyter Notebook с использованием аппаратного ускорения.

**Тест 1:** «Преобразование необработанного звука в нормализованную спектрограмму»

Описание: формирование графика отображения необработанного звука и нормализованной спектрограммы для него (рис. 4.1).

Полученный результат:

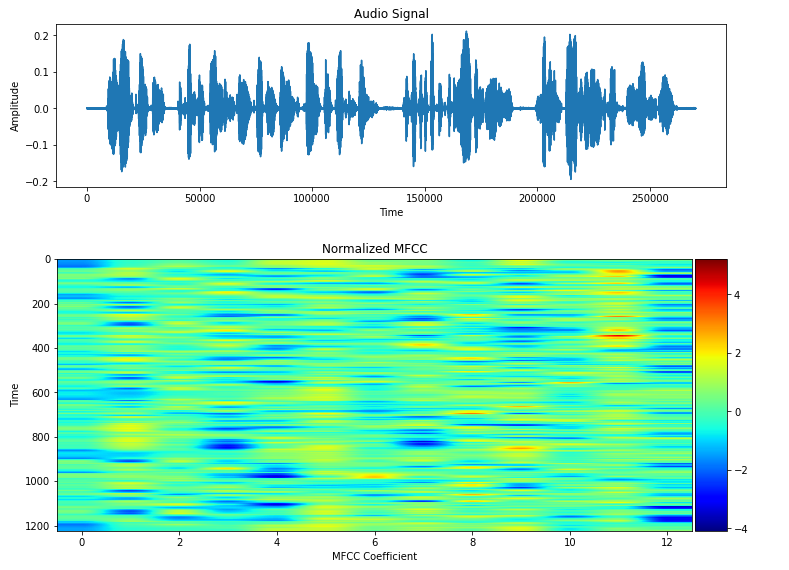


Рисунок 4.1 – График отображения необработанного звука и нормализованной спектрограммы для него

Вывод: сформированы график отображения необработанного звука и нормализованная спектрограмма.

**Тест 2:** «Построение модели»

Описание: построение модели для обучения нейронной сети на основании архитектуры предложенной в главе 2 (рис. 4.2).

Полученный результат:

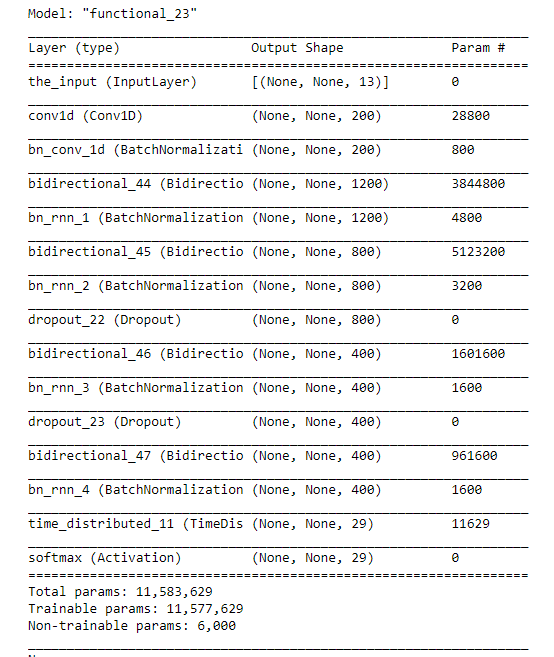


Рисунок 4.2 – Модель, используемая для обучения нейронной сети

Вывод: построена модель для обучения нейронной сети, которая содержит 1D сверточный слой, 4 слоя двунаправленных рекуррентных сетей. Каждый слой сопровождается пакетной нормализацией для уменьшения переобучения. Крайние слои рекуррентной сети сопровождаются исключением для обобщения данных.

**Тест 3:** «Обучение разработанной модели»

Описание: обучить разработанную модель для распознавания речи (рис. 4.3, 4.4). В ходе обучения следует внимательно следить за тем, чтобы с прохождением каждой эпохи функция распространения ошибки loss уменьшалась. Это говорит о том, что процесс обучения модели происходит верно. В таком случае, значение loss на некотором этапе перестанет уменьшатся и будет находится в пределах какого-то определенного значения. Это говорит о том, что сеть начинает переобучение, а значит необходимо прекратить обучение. Экспериментально изменяя такие параметры обучения, как размер мини-выборки, скорость обучения и др. необходимо достичь как можно меньшего значения ошибки loss. В конце каждой эпохи проверяется значение ошибки и раз несколько эпох происходит резервное сохранение модели и её весовых коэффициентов.

Полученный результат:

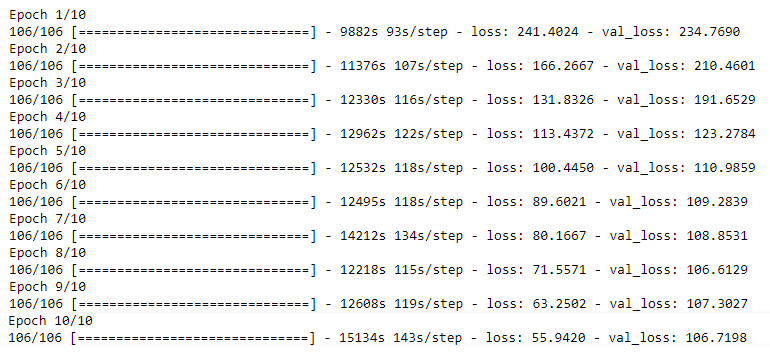


Рисунок 4.4 – Результаты обучения разработанной модели для распознавания речи

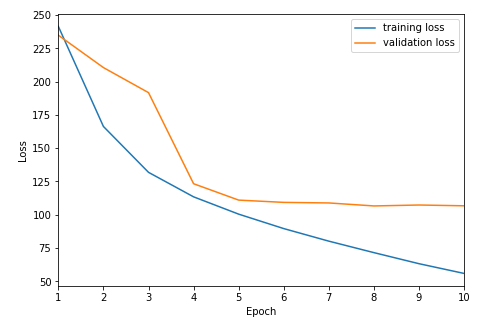


Рисунок 4.4 – Результаты обучения разработанной модели для распознавания речи

Вывод: обучение нейронной сети было остановлено на 10й эпохе, тк значение ошибки для проверочной выборки, начиная с 8 по 10 эпоху перестало уменьшаться. Это означает, что сеть близка к переобучению.

В результате эксперимента была обучена и сохранена нейронная есть для распознавания речи.

**Тест 4:** «Тестирование разработанной модели»

Описание: необходимо протестировать обученную модель для распознавания речи. На рисунках 4.5 и 4.6 представлен интерфейс для выбора файла из существующих на устройстве и для записи нового аудиофайла с помощью микрофона соответственно. Полученный аудиофайл подается на функцию для распознавания голосовых команд и преобразуется в текстовый формат (рис 4.7). Данный аудиофайл также можно повторно прослушать (рис. 4.8).

Полученный результат:

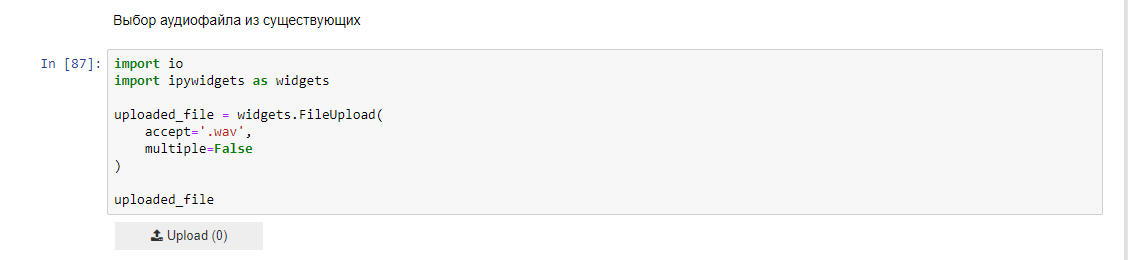


Рисунок 4.5 – Интерфейс для выбора аудиофайла из существующих

 Рисунок 4.6 – Интерфейс для записи аудиофайла с помощью микрофона



Рисунок 4.7 – Результаты тестирования обученной модели для распознавания речи

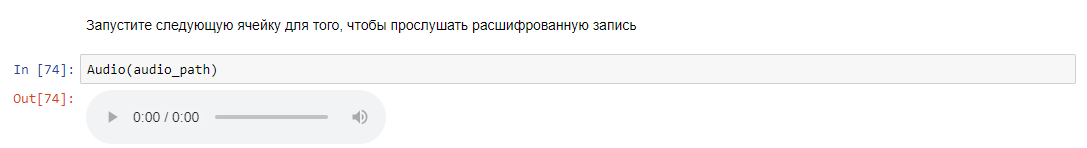


Рисунок 4.8 – Интерфейс для повторного прослушивания аудиофайла

Вывод: в результате эксперимента была протестирован модель для распознавания речи. Она формирует не очень точные предсказания с опечатками и ошибками в словах. Если увеличить объем тестовой выборки, подобрать более подходящие параметры для нейронной сети, можно получить более точные результаты распознавания речи. Но для этого нужны более мощные вычислительные ресурсы.

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта была разработана глубокая нейронная сеть на базе архитектуры Deep Speech 2: End-to-End Speech Recognition [3], обученная на наборе данных Librispeech [4].

Используемый в проекте подход к распознаванию голосовых команд на английском языке дает возможность адаптировать обученную модель нейронной сети распознавать речь на других языках. Таким образом, он является перспективным в задачах распознавания голосовых команд, для применения в различных практических областях (медицина, образование и др.). Испытания показали, что сеть такого типа, можно и дальше улучшать, подбирая необходимые условия и расширяя выборку.

Главным минусом нейросетевых моделей является повышенная временная сложность процесса обучения, высокая зависимость результата от начальной инициализации весовых коэффициентов нейронов, высокие требованиями к мощности вычислительных ресурсов и репрезентативности обучающего множества. Все это обуславливает необходимость наличия определенных навыков в использовании НС при решении практических задач подобного класса.

Программное обеспечение работает под управлением любой популярной операционной системой: Windows, Linux или MacOS с предустановленным интерпретатором Python и некоторыми библиотеками для него.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Антонио Джулли, Суджит Пал, Библиотека Keras – инструмент глубокого обучения: пер. с англ. Слинкин А. А. – М.: ДМК Пресс, 2018 – 294 с.: ил.
2. Why is speech recognition difficult [Электронный ресурс]. – 2020. – Режим доступа: https://www.researchgate.net/publication/228763868\_Why\_is\_ speech\_recognition\_difficult – Дата доступа: 12.12.2020.
3. Deep Speech 2: End-to-End Speech Recognition in English and Mandarin [Электронный ресурс]. – 2020. – Режим доступа: https://arxiv.org/pdf/1512.02595v1.pdf – Дата доступа: 12.12.2020.
4. Librispeech [Электронный ресурс]. – 2020. – Режим доступа: http://www.openslr.org/12 – Дата доступа: 12.12.2020.