МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ НАУК
Кафедра программного обеспечения

РЕКОМЕНДОВАНО К ЗАЩИТЕ В ГЭК
Заведующий кафедрой, к.т.н., доцент
_____М. С. Воробьева
02.07. 2021г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
магистерская диссертация

РАЗРАБОТКА И РЕАЛИЗАЦИЯ ТЕХНОЛОГИИ ФОРМИРОВАНИЯ ТЕКСТОВОГО АНАЛОГА ЗВУКОВОЙ ЗАПИСИ РАЗГОВОРА

02.04.03 Математическое обеспечение и администрирование информационных систем
Магистерская программа «Разработка технологий Интернета вещей и больших данных»

Выполнил работу
студент 2 курса
очной формы обучения — Попов Алексей Олегович

Научный руководитель
доцент, к.ф.-м.н. — Ступников Андрей Анатольевич

Рецензент
Web-разработчик
ООО «ГеоНавТех» — Пиджаков Святослав Игоревич

Тюмень
2021

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Для успешной подготовки и защиты выпускной квалификационной работы обучающимся использовались средства и методы физической культуры и спорта с целью поддержания должного уровня физической подготовленности, обеспечивающую высокую умственную и физической работоспособность. В режим рабочего дня включались различные формы организации занятий физической культурой (физкультпаузы, физкультминутки, занятия избранным видом спорта) с целью профилактики утомления, появления хронических заболеваний и нормализации деятельности различных систем организма.

В рамках подготовки к защите выпускной квалификационной работы автором созданы и поддерживались безопасные условия жизнедеятельности, учитывающие возможность возникновении чрезвычайных ситуаций.

Цифровая обработка аудио сигналов на сегодняшний момент достаточно широко используется в разных сферах деятельности. Благодаря цифровой обработке аудио сигналов стало возможным выделение отдельные звуков из аудиодорожки, очистка шумов, наложение фильтров, преобразование речи в текст, а также комбинирование этих звуковых сигналов и создание новых звуковых дорожек.

По существующим технологиям распознавания текста по имеющейся записи человеческой речи, такой текст формируется не персонифицированным по говорящим людям и для человека со слабым слухом сложно определить, что говорил тот или иной человек.

Также структурированная в виде текста речь человека позволяет быстрее устанавливать соответствие текста и говорящего человека и осуществлять поиск необходимой информации. Такая технология позволит автоматизировать процесс создания субтитров для фильмов в режиме реального времени и для разных языков с обозначением имени говорящего.

Такая технология преобразования речи в текст с распределением по голосам помогла бы автоматизировать ведение текстовых протоколов совещаний или слушаний в режиме реального времени.

К примеру, во время суда происходит ведение протокола заседания, где записываются слова судьи, ответчика, истца и других участников заседания. Вся эта работа выполняется секретарем, в случае если имелся бы такой инструмент автоматического ведения протокола, то это позволило бы сэкономить время секретарю, а также позволяет устранить возможность преднамеренного или непреднамеренного искажения вручную фиксируемых данных.

Цель данной работы: программная реализация и исследование подходов к созданию структурированного текстового варианта записи разговора с разделением фраз по говорящим людям.

Задачи:

- Изучение специфики и технологий работы с аудиоинформацией.
- Разбиение и разметка аудиодорожек по говорящим с сохранением порядка следования.
- Реализация модулей для персонализации речи.
- Анализ эффективности моделей и инструментов обработки аудиосигналов.

Для подготовки и защиты выпускной квалификационной работы использовались поиск, анализ информации, системный подход для решения поставленных задач; приемы критического анализа проблемных ситуаций, а также средства и методы саморазвития и самореализации; методики межкультурного взаимодействия; умение расставлять приоритеты собственной деятельности при работе в общем проекте в соответствии с командной стратегией для достижения поставленной цели. Формулирование выводов по итогам проведенной работы осуществлялись с учетом применения современных коммуникативных технологий (в том числе на иностранном языке) для представления результатов на академических, профессиональных, экспертных ИТ-мероприятиях.

ГЛАВА 1. МАТЕМАТИЧЕСКИЕ МОДЕЛИ ЦИФРОВОЙ ОБРАБОТКИ СИГНАЛОВ

1.1. ВЫДЕЛЕНИЕ ЗВУКОВЫХ ЧАСТОТ. ПРЕОБРАЗОВАНИЕ ФУРЬЕ. СПЕКТР.

Человеческая речь представляет собой набор последовательных звуков. Звук в свою очередь — это суперпозиция (наложение) звуковых колебаний (волн) различных частот. Звуковая волна имеет два параметра для описания — амплитуда и частота.

Преобразование Фурье – это математический аппарат для разложения сигналов на сумму синусоидальных колебаний с разной частотой и амплитудой. Например, если сигнал $x(t)$ непрерывный и бесконечный по времени, то его можно представить в виде интеграла Фурье:

$$x(t) = \int_0^\infty X_\omega \cos(\omega t + \varphi)\, d\omega \qquad (1.1.1)$$

Интеграл Фурье собирает сигнал $x(t)$ из бесконечного множества синусоидальных составляющих всевозможных частот ω, имеющих амплитуды X_ω и фазы φ_ω [19].

С практической точки зрения интерес представляет анализ конечных по времени звуков. Спектр аудиосигнала меняется во времени, поэтому при спектральном анализе представляют интерес отдельные короткие фрагменты сигнала. Для анализа таких фрагментов цифрового аудиосигнала существует дискретное преобразование Фурье по формуле:

$$X_k = \sum_{n=0}^{N-1} X_n e^{-\frac{2\pi i}{N} kn}, \qquad (1.1.2)$$

где N – количество обрабатываемых сигналов,

X_n- измеренное значение сигнала,

X_k – N комплексных амплитуд синусоидальных сигналов, слагающий исходный сигнал,

k – значение частоты сигнала.

Дискретное преобразование Фурье позволяет преобразовать аналоговые сигналы для вычислительной обработки в цифровые в виде

частотно-амплитудного спектра (Рис 1.1.1). Красные столбцы обозначают мощность сигнала соответствующей частоты, а зеленная линия отображает предполагаемое теоретическое распределение частот.

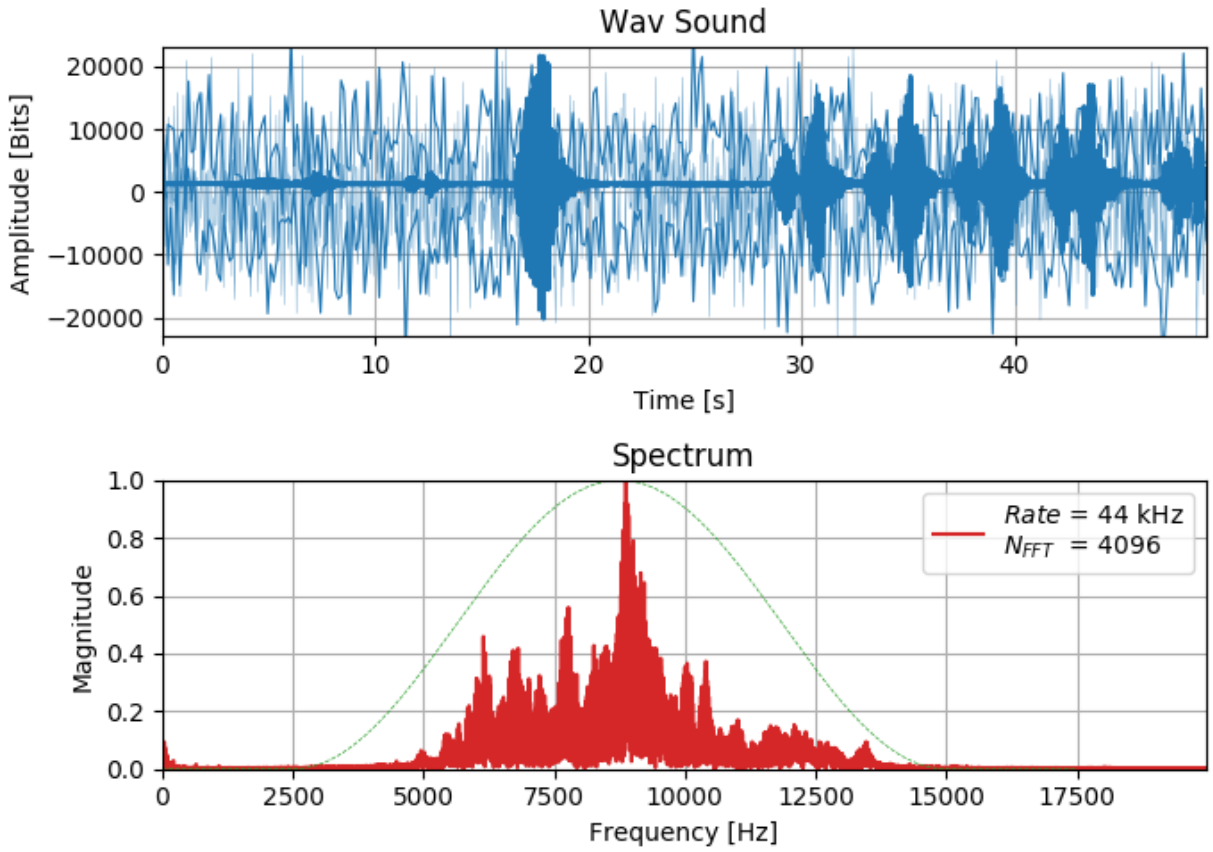


Рис. 1.1.1. Преобразование в частотно-амплитудный спектр

Поскольку аналоговый сигнал состоит из бесконечного непрерывного во времени множества точек-значений амплитуды, в процессе измерения можно выделить из него лишь конечный ряд значений в дискретные моменты времени, т.е. выполнить квантование по времени (рис.1.1.2) [25].

Как правило, значения-отсчёты берутся через небольшие равные временные промежутки, то есть с определённой частотой, например, 16000 или 22000 Гц. Однако в общем случае дискретные отсчёты могут идти и неравномерно, но это усложняет математический аппарат анализа, поэтому на практике обычно не применяется.

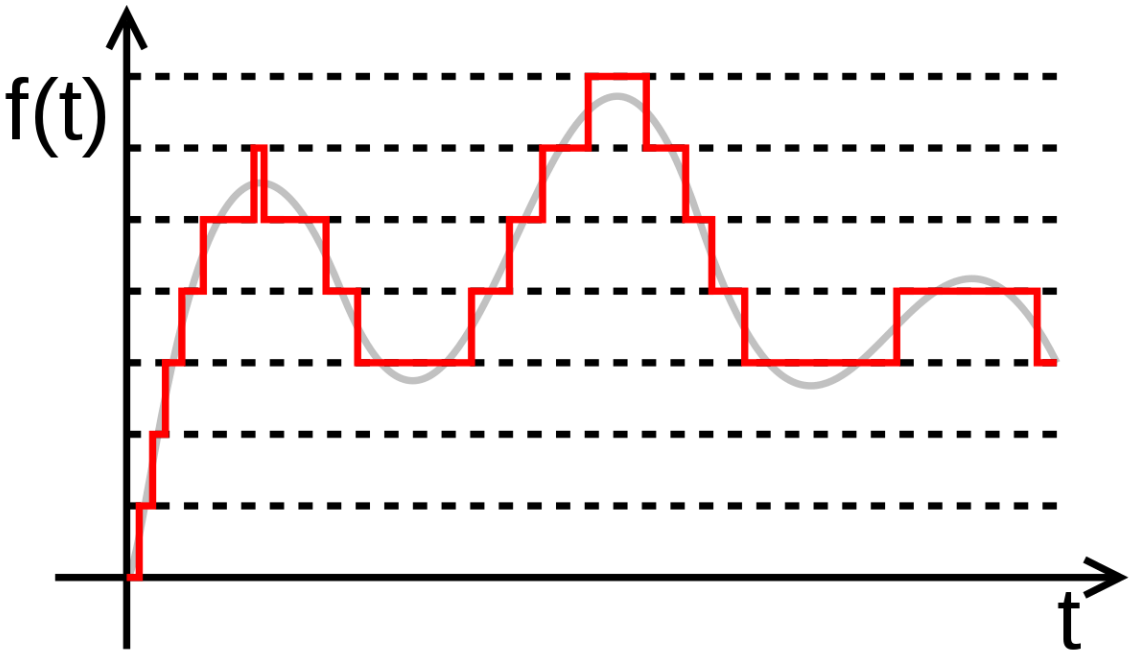


Рис. 1.1.2 График квантования

Согласно теореме Котельникова-Найквиста-Шеннона: аналоговый периодический сигнал, имеющий конечный (ограниченный по ширине) спектр, может быть однозначно восстановлен без искажений и потерь по своим отсчётам, взятым с частотой, большей или равной удвоенной верхней частоте спектра (называемой частотой дискретизации или Найквиста) [29].

Говоря шире, теорема Котельникова утверждает, что непрерывный сигнал $x(t)$ можно представить в виде интерполяционного ряда:

$$x(t) = \sum_{k=-\infty}^{\infty} x(k\Delta) sinc[\frac{\pi}{\Delta}(t - k\Delta)] \qquad (1.1.3)$$

Где $sinc(x) = \sin(x)/x$ - функция sinc. Интервал дискретизации удовлетворяет ограничениям $0 < \Delta \leq 1/2f_c$. f_c – максимальная частота, которая ограничена спектром реального сигнала. Мгновенные значения данного ряда есть дискретные отсчёты сигнала $x(k\Delta)$.

Также дискретизация происходит не только по времени, но и по уровню значений амплитуды, поскольку компьютер способен манипулировать лишь ограниченным множеством чисел. Это также вносит небольшие погрешности.

При дискретном преобразовании аудио сигнала выделяется короткий кадр (интервал) композиции, состоящий из дискретных отсчётов, к которому применяется преобразование Фурье. После использования преобразования, на выходе получается массив комплексных чисел, содержащий информацию об амплитудном и фазовом спектрах анализируемого кадра. Причём спектры также являются дискретными с шагом равным:

$$h = \frac{f_c}{N} \tag{1.1.4}$$

Где f_c – частота дискретизации, N – количество отсчетов.

Чем больше берется отсчётов для анализа, тем точнее получается разрешение по частоте. При увеличении числа отсчётов не меняя частоту дискретизации увеличивается анализируемый временной интервал. У дискретного преобразования Фурье имеется модификация под названием быстрое преобразование Фурье.

FFT (fast Fourier transform) – алгоритм быстрого вычисления дискретного преобразования Фурье. Его применение позволило анализировать спектр звуковых сигналов в реальном времени [26].

Данный алгоритм основан на рекурсивном подходе обеспечивает получение результата гораздо быстрее чем использование обычного дискретного преобразования, однако требуется отдавать на вход алгоритму размер кадра равным 2^n. Время работы быстрого преобразования Фурье сводится к O(N*log(N)).

Также для увеличения скорости анализа динамики звуковых сигналов, размер кадра используемого в преобразовании Фурье берут равным не более 1024 элементов. Однако точность получаемых данных уменьшается в зависимости от размера кадра (чем меньше кадр – тем меньше точность получаемых данных).

Чтобы избежать искажения данных, получаемых в спектре аудиодорожки, применяется процедура умножения каждого элемента обрабатываемого кадра на особую весовую функцию («окно»). В результате

выделяется наиболее важная часть кадра с плавным затуханием амплитуд на его краях.

Такой подход позволяет достичь более лучших результатов при использовании преобразования Фурье. Само по себе преобразование ориентирована на бесконечно повторяющийся сигнал. [13]. Соответственно, кадр должен стыковаться сам с собой и как можно более плавно. Окон существует огромное множество. Лучше всего использовать окно Хэмминга:

$$w(n) = 0.53836 - 0.46164\cos(\frac{2\pi n}{N-1}) \quad (1.1.5)$$

где n — порядковый номер элемента в кадре, для которого вычисляется новое значение амплитуды, N —длина кадра (количество значений сигнала, измеренных за период). Применяют оконную функцию по следующей формуле:

$$F(m,\omega) = \sum_{n=-\infty}^{\infty} f[n]w[n-m]e^{-j\omega n} \quad (1.1.6)$$

где w — некоторая оконная функция. Например, окно Хэмминга, которое уменьшает размытие спектра за счет ухудшения частотного разрешения. Существуют и другие оконные функции, простейшее из них – прямоугольное: это константа 1, не меняющая сигнала (рис. 1.1.3). Окно Кайзера или окно Блэкмана также уменьшают размытия спектра, но в разной степени.

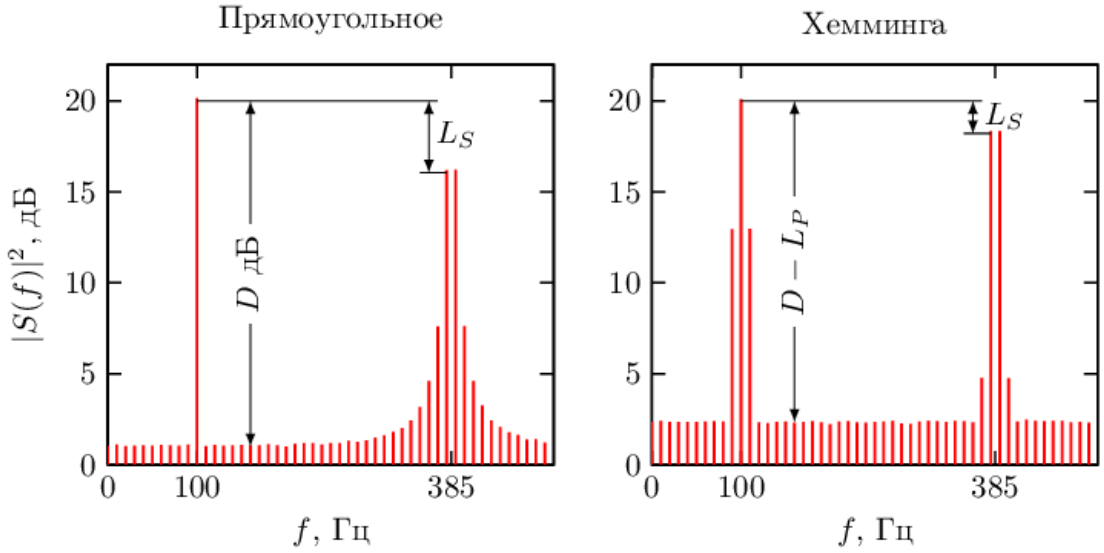


Рис. 1.1.3 Применение окна Хэмминга

Иногда для работы со звуком требуется визуализация изменений спектра на всем звуковом ряде. Такое представление сигнала называется спектрограммой. Для её построения применяется оконное преобразование Фурье: спектр вычисляется от последовательных окон сигнала, и каждый из этих спектров образует столбец в спектрограмме.

По горизонтальной оси спектрограммы откладывается время, по вертикальной – частота. Если менять размер окна FFT, становится видно, как меняется частотное и временное разрешение спектрограммы. При увеличении окна гармоники становятся тоньше, и их частота может быть определена более точно. При уменьшении размера окна наблюдается обратный эффект. Полученную спектрограмму впоследствии используют для получения мел-кепстральных коэффициентов.

1.2. ВЫДЕЛЕНИЕ МЕЛ-КЕПСТРАЛЬНОЙ МОДЕЛИ ЗВУКА

Мел – единица высоты звука, основанная на его восприятии человеческими органами слуха. Как известно, АЧХ (Амплитудно-частотная характеристика) человеческого уха даже отдаленно не напоминает прямую, и амплитуда – не является достаточно точной мерой громкости звука [21].

Воспринимаемая человеческим слухом высота звука нелинейно зависит от его частоты (рис. 1.2.1).

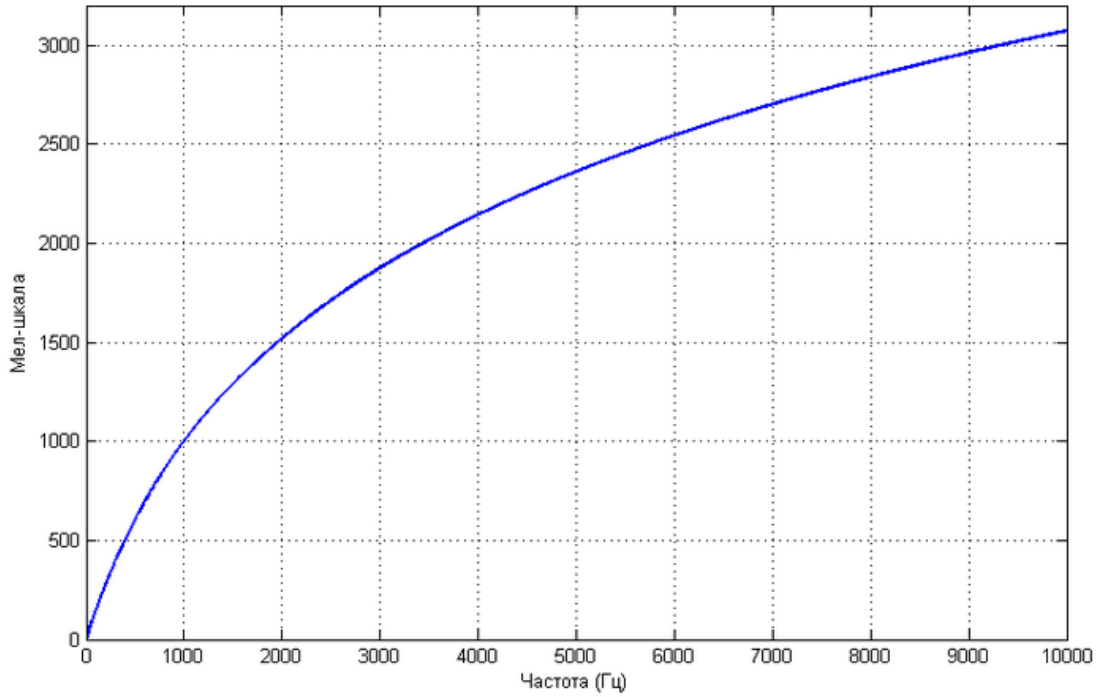


Рис.1.2.1. График мел-шкалы

Такая зависимость не рассчитана на большую точность, но, с другой стороны, описывается простой формулой (мел-формула):

$$m = 1125\ln(1 + \frac{f}{700}) \quad (1.2.1)$$

f – значение частоты.

m – значение единицы измерения мел.

Подобные единицы измерения (Мел) часто используют при решении задач распознавания, так как они позволяют приблизиться к механизмам человеческого восприятия, которое в настоящий момент времени лидирует среди известных систем распознавания речи.

Для получения сигнатуры человеческого голоса в виде набора мел-кепстральных коэффициентов используются мел-окна. В первую очередь получают спектр человеческого голоса с помощью преобразования Фурье. После этого полученный спектр преобразуют по мел-формуле. На преобразованном графике выделяют равноудаленные друг от друга точки и выполняют обратное преобразование в спектрограмму по следующей формуле:

$$F = 700 * (e^{\frac{M}{1127}} - 1) \quad (1.2.2)$$

Полученные точки на спектре являются опорным, их используют для расположения фильтра мел-окон (рис. 1.2.2) [27]. Мел-окна по своей сути являются треугольной оконной функцией, позволяющей просуммировать количество энергии на заданных диапазонах частот, что позволяет получить мел-спектральные коэффициенты. Треугольная функция имеет следующую формулу:

$$w[n] = 1 - \left| \frac{n - \frac{N}{2}}{\frac{L}{2}} \right|, \quad 0 \leq n \leq N \quad (1.2.3)$$

где L – может быть равно N, N+1 или N+2,

N – размер кадра дискретного преобразования Фурье.

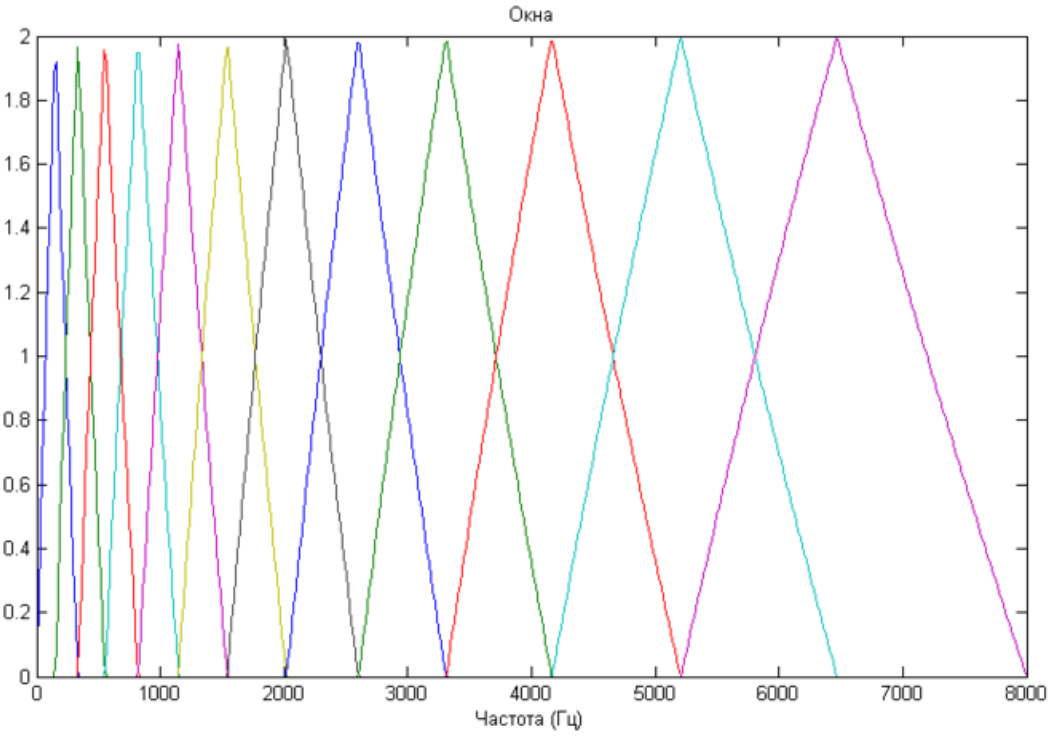


Рис.1.2.2. Мел-треугольные окна

Чем больше порядковый номер треугольного окна, тем шире будет основание фильтра будет. Это связано с тем, что разбиение диапазона частот, обрабатываемых фильтрами, происходит на мел-шкале.

Перемножая вектор спектра сигнала и мел-окна, в результате рассчитывается энергия сигнала для каждого окна. В итоге получаем вектор,

который называют мел-частотными спектральными коэффициентами. Однако это не мел-кепстральные коэффициенты.

Для получения мел-кепстральных коэффициентов, набор мел-частотных спектральных коэффициентов возводят в квадрат, логарифмируют и с помощью дискретного косинусного преобразования (формула 1.2.4) получают «кепстр» этих коэффициентов.

$$C[l] = \sum_{m=0}^{M-1} S[m] * \cos(\pi * l * \frac{m+\frac{1}{2}}{M})), 0 \leq l \leq M \qquad (1.2.4)$$

где M – количество мел-кепстральнных коэффициентов.

Смысл дискретного косинусного преобразования состоит в том, чтобы “сжать” полученные результаты, повысив значимость первых коэффициентов и уменьшив значимость последних. Часто дискретное косинусное преобразование применяют в алгоритмах сжатия информации с потерями [16].

Такое преобразование применяется для каждого фрейма для получения наборов мел-кепстральных коэффициентов для каждого кадра на аудиодорожке. Все наборы агрегируют для получения единого вектора сигнатуры человеческого голоса.

Полученный вектор значений своего рода является сигнатурой человеческого голоса. [20]

Наборы векторов сравниваются через евклидово расстояние:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \qquad (1.2.5)$$

Чем меньше получаемое расстояние между векторами, тем вероятнее голоса являются схожими. [14]

1.3. ИДЕНТИФИКАЦИЯ ЧЕЛОВЕКА ПО ГОЛОСУ ПРИ ПОМОЩИ НЕЙРОННЫХ СЕТЕЙ.

Существует другой подход по идентификации человеческой речи, данный подход подразумевает использование методов машинного обучения и нейронных сетей [18].

Сама по себе нейронная сеть – это некая математическая модель, представляющая собой систему соединенных и взаимодействующих между собой простых процессоров (искусственных нейронов) определенным образом связанных друг с другом и внешней средой с помощью связей, каждая из которых имеет определённый коэффициент, на который умножается поступающее через него значение (эти коэффициенты называют весами) [24]. С точки зрения машинного обучения, нейронная сеть представляет собой частный случай методов распознавания образов [23].

Как раз сигнатура человеческой речи является таким образом, который можно классифицировать. В итоге задача идентификации человеческой речи сводится к задаче классификации мел-кепстральных коэффициентов.

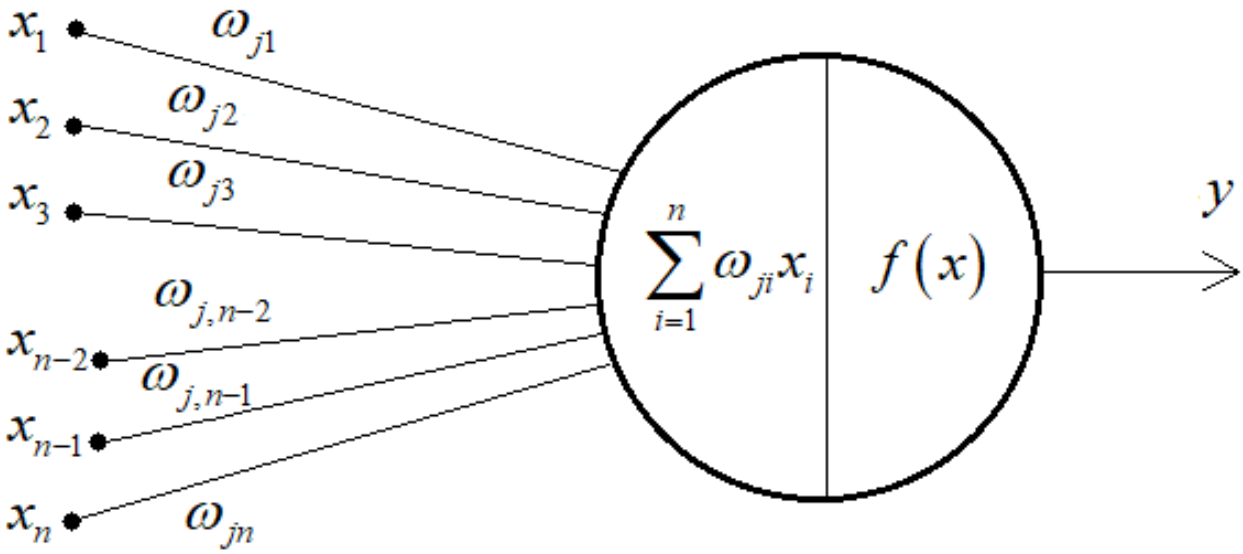


Рис 1.3.1 Пример нейронной сети

Рассмотрим на простом примере метод идентификации речи через нейронную сеть (рис 1.3.1).

Дана аудиодорожка, в которой записан человеческий голос. Для начала из этой аудиодорожки извлекаются признаки, которые будут использоваться для входного слоя нейронной сети. В данном случае этими признаками являются мел-кепстральные коэффициенты.

Признаки – это набор чисел, характеризующих оратора – вектор в многомерном пространстве. Задача классификатора заключается в том,

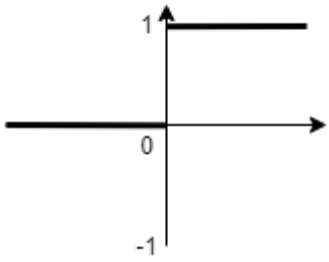
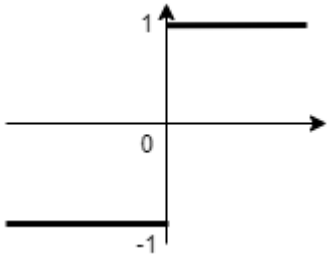
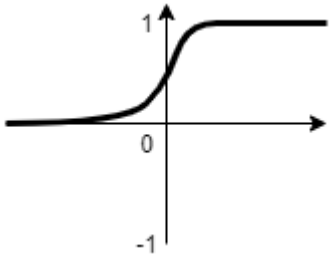
чтобы построить функцию отображения этого многомерного пространства в конечное множество. Другими словами, задача классификации состоит в получении значения, описывающего меру схожести.

Пусть $x_1,x_2,...,x_n$, – вектор входных данных (мел-кепстральных коэффициентов), которые поступают на нейронную сеть. Каждый элемент вектора умножается на соответствующий вес $w_{j1}w_{j2},...,w_{jn}$, следующего слоя после этого используется функция активация и, благодаря которой выдается выходное значение нейронного сигнала, в зависимости от результата взвешенной суммы входов и порогового значения [17].

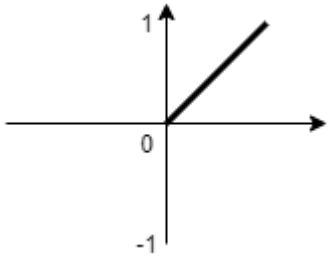
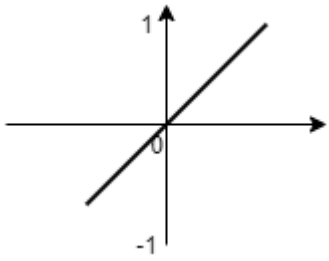
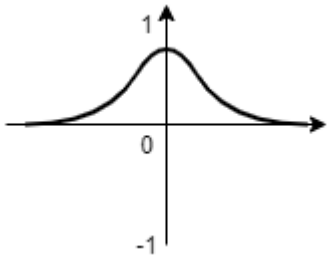
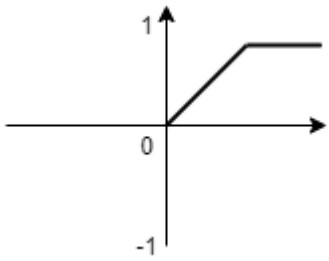
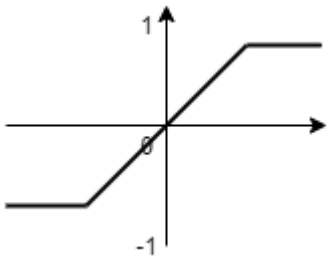
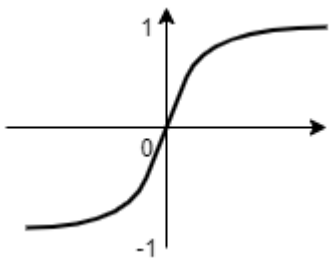
Существует множество функций активации (табл. 1.3.1). Наиболее часто использующимися функции являются ReLu (полулинейная) и Сигмоидальная (логистическая).

Таблица 1.3.1

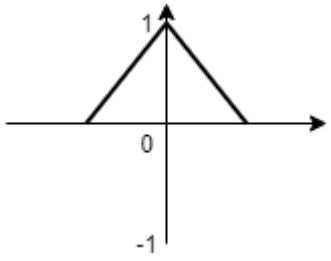
Функции активации

Название	Формула	График
Пороговая	$f(x) = \begin{cases} 0, x < 0 \\ 1, x \geq 1 \end{cases}$	
Знаковая (сигнатурная)	$f(x) = \begin{cases} 1, x > 0 \\ -1. x \leq 0 \end{cases}$	
Сигмоидальная (логистическая)	$f(x) = \frac{1}{1 + e^{-x}}$	

Продолжение таблицы 1.3.1

Полулинейная	$f(x) = \begin{cases} x, x > 0 \\ 0, x \leq 0 \end{cases}$	
Линейная	$f(x) = x$	
Радиальная базисная (гауссова)	$f(x) = e^{-x^2}$	
Полулинейная с насыщением	$f(x) = \begin{cases} 0, x \leq 0 \\ x, 0 < x < 1 \\ 1, x \geq 1 \end{cases}$	
Линейная с насыщением	$f(x) = \begin{cases} -1, x \leq -1 \\ x, 0 < x < 1 \\ 1, x \geq 1 \end{cases}$	
Гиперболический тангенс (сигмоидальная)	$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	

Продолжение таблицы 1.3.1

Треугольная	$f(x) = \begin{cases} 1 - \lvert s \rvert & \lvert x \rvert \leq 1 \\ 0 & \lvert x \rvert > 0 \end{cases}$	

Сами по себе эти две функции активации нелинейны, а также их комбинация тоже нелинейна. При помощи функции сигмоиды можно привести значение каждого нейрона к одной из сторон функции этой кривой что позволит улучшить качество модели при решении задач классификации.

Функция активации ReLu позволяет «разрядить» сеть, то есть некоторые нейроны не будут активированы для расчета, что сделает сеть более эффективной [30].

На этапе процесса обучения происходит настройка параметров нейронной сети. Различают два этапа обучения нейросетей:

- Прямой проход – при котором делается предсказание ответа. Для этого выполняется расчет в нейронной сети для каждого слоя, где количество нейронов и функции активации могут быть разными. Дойдя до выходного слоя, происходит расчет вероятности принадлежности к классу, при этом важно, чтобы количество нейронов в выходном слое было равно количеству классов.
- Обратный проход – обеспечивает расчет параметров нейронной сети для увеличения точности предсказания. В этом случаем применяется функция потерь необходимая, как способ количественно оценить, насколько сеть «хороша», чтобы можно было улучшить нейронную сеть. Чем меньше значение функции потерь, тем лучше предсказывает нейронная сеть.

Также для улучшения работы нейронной сети используют оптимизатор. Это алгоритм, используемый для незначительного изменения

параметров, таких как веса и скорость обучения, чтобы модель работала правильно и быстро.

Adam — один из самых эффективных алгоритмов оптимизации в обучении нейронных сетей. Данный алгоритм обеспечивает незначительное обновление весов для типичных признаков [22].

Адам представляет собой объединение двух других методов оптимизации машинного обучения:

- Адаптивный Градиентный Алгоритм (AdaGrad), улучшает производительность машинной модели с проблемами разреженных градиентов и способствует сохранению оптимальной скорости обучения (например, при проблемах с обработкой речи и распознаванием визуальных образов).
- Среднеквадратичное распространение (RMSProp), аналогично AdaGrad также способствует сохранению оптимальной скорости обучения по каждому параметру модели и адаптируется, используя как основу среднее значение последних величин градиентов для изменения веса. То есть, алгоритм хорошо справляется с задачами, обрабатываемыми в реальном времени (например, с шумом).

Параметры бета1 и бета2 в алгоритме Adam (см. параграф 2.3) управляют скоростью затухания этих скользящих средних. Параметр Epsilon задается для того, чтобы избежать деление на 0, а также параметр learning_rate задается чтобы модулировать изменением скорости обучения оптимизатора с течением времени.

Система распознавания работает в двух режимах: в режиме регистрации и режиме идентификации. Другими словами, необходимо иметь пример голоса.

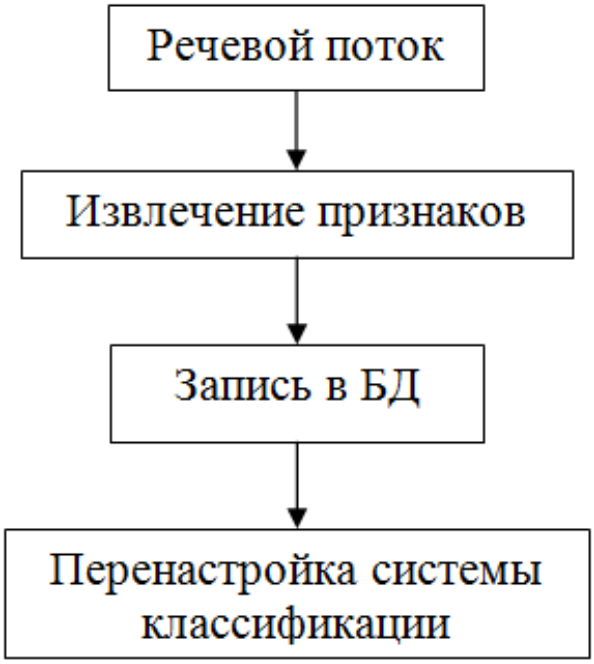


Рис 1.3.3 Схема режима регистрации

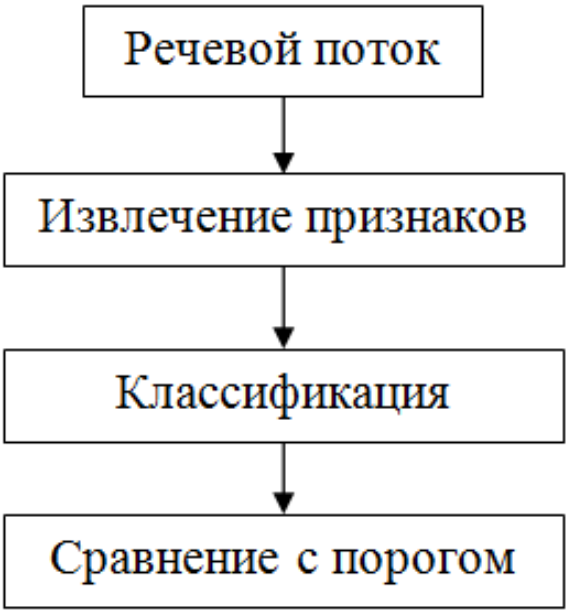


Рис 1.3.4 Схема режима идентификации

На иллюстрациях (рис 1.3.3, 1.3.4) представлена общая схема работы системы в каждом из режимов. Эти режимы весьма похожи. Обоим для работы необходимо захватить речевой аудио поток и вычислить его

основные признаки. Отличие же состоит в том, что же делать с этими признаками. При регистрации их так или иначе необходимо как-то запомнить для использования в будущем. Ведь гораздо эффективнее работать с основными признаками, чем с исходными сырыми данными.

При идентификации ничего сохранять нельзя, так как система на данном этапе не имеет обратной связи и не может достоверно знать принадлежность голоса [29].

1.4. ВЫБОР ИНСТРУМЕНТОВ ДЛЯ ОБРАБОТКИ АУДИОСИГНАЛОВ

В ходе работы рассматривались программные библиотеки по обработке аудио для разных языков программирования.

Для языка C# наиболее популярной технологией для работы с аудиоинформацией является библиотека NAudio [7] со множеством функций извлечения данных с аудиодорожки и поддержкой множества аудио форматов таких как:

- WAV.
- AIFF.
- MP3.
- G.711 mu-law and a-law.
- ADPCM, G.722, Speex.
- WMA, AAC, MP4 и многие другие.

Имеет следующие возможности для работы с аудио потоками:

- Создание сигнальных цепей.
- Проверка уровней выборки для измерения или визуализации сигналов.
- Обработка блоков выборок через БПФ для измерения.
- Обработка задержки сигнала, а также зацикливание или исчезновение звука.

- Обработка через эквалайзер с фильтром BiQuad (эквалайзер позволяет обрабатывать низкие и высокие частоты, а также пиковые частоты и т. д.).
- Изменение высоты звука с помощью фазового кодировщика голоса.

Однако простого способа получения мел-кепстральных коэффициентов в нем нет, и использование методик машинного обучения было бы затруднительно на этом языке программирования.

Для языка программирования Python был рассмотрен набор инструментов librosa [6], который представляет собой средства для анализа музыки и аудио. Благодаря данному инструменту можно извлечь множество признаков из аудиофайлов, такие как: темп, бит, такт, интервал, ритм, частоту дискретизации, а также один из важнейших признаков, используемых в этой работе – мел-кепстральные коэффициенты. Поддерживает множество разных аудио форматов что и библиотека NAudio. Впоследствии был выбран данный пакет и язык программирования Python.

В ходе дальнейшего поиска инструментов для работы с аудиоинформацией также найдена библиотека pydub [8]. При помощи этого инструмента возможно разделение аудиодорожки на части, содержащие только речь человека относительно пауз (тишины) в диалоге используемой аудио дорожки.

Что касается преобразования речи в текст, для этой цели используется библиотека SpeechRecognition [10]. Она позволяет подключиться ко множеству ресурсов по типу Google или Microsoft, имея соответствующий API ключ, и использовать ресурсы для преобразования речи в текст. Также имеется режим без подключения к интернету, однако для этого требуется дополнительная установка пакета.

Для реализации описанных ниже методов машинного обучения используется библиотека Keras [2]. Это мощный отраслевой фреймворк для среды программирования Python, созданный на основе TensorFlow 2.0. Имеет

интуитивно понятный API и содержит обширную документацию и руководства для разработчиков.

Преимущества в использовании Keras:

- Простота в использования и высокая скорость развертывания
- Качественная документация и поддержка от сообщества
- Поддержка разных движков и модульность (Поддержка Tensorflow, Theano и CNTK)
- Натренированные модели
- Поддержка нескольких GPU

Благодаря данной программной библиотеки, можно без больших затрат сил и времени реализовать простые модели машинного обучения.

1.5. СИСТЕМА УПРАВЛЕНИЯ ЭКСПЕРИМЕНТАМИ МАШИННОГО ОБУЧЕНИЯ COMET ML

Comet.ml – это система управления экспериментами по машинному обучению. Данная система предоставляет следующие возможности в рамках обработки процедур с большими данными и машинным обучением:

- Представление графиков обучения модели (графики точности, потерь)
- Логирование различных типов данных (текст, изображения, аудиозаписи, видеозаписи)
- Сравнение экспериментов (тестов) проекта
- Возможность поделиться результатами эксперимента с другими людьми
- Возможность быстрой интеграции – достаточно добавить одну строку кода для старта отслеживания экспериментов.
- Развертывание везде, где запускается код, с любой библиотекой машинного обучения и для любой задачи машинного обучения.
- Создание собственных визуализации на основе экспериментов и данных моделей или использование предоставленных сообществом.

Comet предоставляет автономную облачную платформу для мета-машинного обучения, позволяющую специалистам по обработке данных, а также командам отслеживать, сравнивать, объяснять и оптимизировать эксперименты и модели [1].

При поддержке тысяч пользователей и нескольких компаний из списка Fortune 100, Comet предоставляет аналитические данные и данные для создания более точных моделей искусственного интеллекта, одновременно повышая продуктивность, сотрудничество и прозрачность для всех команд.

В связке с программной библиотекой Keras система Comet может автоматически регистрировать:

- Описание модели и графиков.
- Шаги и эпохи.
- Показатели (к примеру, потери и точность).
- Гиперпараметры.
- Конфигурация оптимизатора.
- Количество обучаемых параметров.
- Гистограммы весов и смещений.
- Гистограммы для функций активаций.
- Гистограммы для градиентов.

Comet.ml регистрирует эксперимент с помощью обратного вызова, выполняемого при запуске метода model.fit() в Keras. Для дополнительного логирования различных типов данных вызываются соответствующие методы, в случае для логирования аудиодорожек вызывается метод log_audio().

Для использования Comet Ml достаточно добавить несколько строчек кода:

```
from comet_ml import Experiment
experiment = Experiment(
    api_key="****** ",
    project_name="vkr",
    workspace="isaac30",
)
```

Листинг 1.5.1 Код подключения системы Comet ML к проекту

Первым делом импортируется класс Experiment из библиотеки comet_ml. После создается экземпляр этого класса, в котором указываются следующие параметры:

- Api_key – необходимый ключ для того чтобы подключиться к системе Comet ML и передавать полученные данные.
- Project_name – имя проекта, в котором будут создаваться новые эксперименты.
- Workspace – Рабочее пространство содержащее проекты в системе Comet ML

После добавления этих строчек кода, Comet ML сможет автоматически начать регистрировать данные.

Для того чтобы не захватывать лишние данные используют метод experiment.end(), если необходимо закончить эксперимент до выполнения всего цикла программы [4].

ГЛАВА 2. ВАРИАНТЫ ПРОГРАММНОЙ РЕАЛИЗАЦИИ РАЗДЕЛЕНИЯ ГОЛОСОВ

2.1. ПОДГОТОВКА АУДИОДАННЫХ

В работе рассмотрены два способа разделения аудиоданных на голоса. В первом подходе (рис. 2.2.1) для определения разницы голосов применяется сравнение мел-кепстральных коэффициентов.

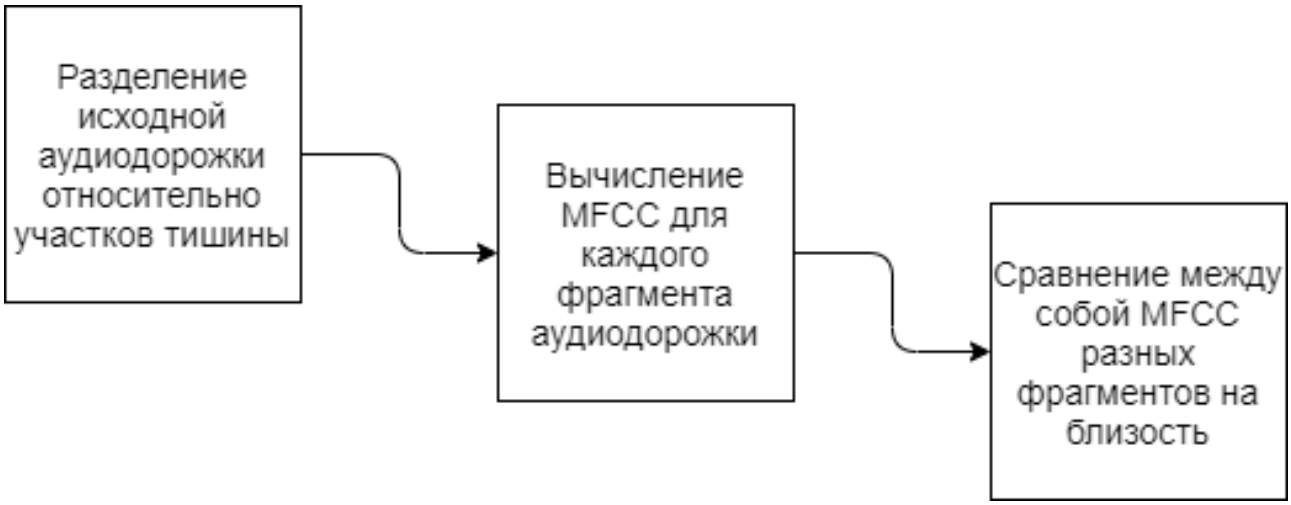


Рис 2.2.1 Этапы работы программы

В поступающих на вход аудиодорожках заранее неизвестно, что говорят люди, сколько посторонней информации в виде тишины, шумов присутствуют на аудиодорожке.

Входная аудиодорожка имеет формат wav, поэтому в первую очередь у объекта AudioSegment используется метод from_wav библиотеки pydub для загрузки аудиофайла в память программы.

Необходимо избавиться от ненужной информации в виде тишины, т.к. для анализа голоса она не нужна. Используя метод split_on_silence, в который передается сам загруженный файл с аудиоданными, устанавливается первый параметр метода в виде минимального порога тишины в минимум 500 миллисекунд. Если установить меньше, можно на выходе получить аудиофайлы с одним словом или даже с одной буквой.

Последний параметр задает верхнюю границу для тишины, для того чтобы исключить помехи, например, при записи голоса с фоновым шумом.

В результате работы метод формирует файлы аудиоданных, в данном случае набор записанных предложений с разными голосами. Для дальнейшей работы файлы сохраняются на жесткий диск компьютера.

В качестве исходных данных для моделей машинного обучения был найден американско-английский набор данных, созданный компанией Surfingtech (www.surfing.ai) [3]. Этот датасет содержит в себе высказывания 10 людей, записанных на телефон в полной тишине. Каждый говорящий имеет около 350 аудиозаписей с точной транскрипцией.

Данный набор аудиозаписей является подмножеством гораздо большего набора данных (около 1000 часов), который был записан при тех же условиях, что и представленные данные.

Представленные аудиоданные имеют следующий шаблонный вид: «m0002_us_m0002_00036.wav», где m/f идентифицирует пол говорящего, а «0002» – его номер (числовой идентификатор). Набор цифр «00036» является номером записи относительно идентификатора. И обозначение «us» говорит о том, что запись на английском языке. Все имеющиеся файлы представлены в формате wav с частотой дискретизации, равной 16000 Герц.

Изначально, аудиозаписи не относятся явно к какому-либо классу. Для этого необходимо для каждой аудиозаписи взять идентификатор говорящего в качестве класса, к которому относится аудиозапись, и сформировать следующий список пар: путь к аудиозаписи – идентификатор.

Полученный список обрабатывается, прежде чем передавать для обучения. Для каждой аудиозаписи берутся кадры в размере 4096 для преобразования с частотой дискретизации 16кГц, вычисляются мел-кепстральные коэффициенты в количестве 34 штук и для каждого кадра происходит агрегация данных. Получивший набор имеет вид: набор мел-кесптральных коэффициентов – идентификатор аудиозаписи.

Так как нейронная сеть взаимодействует только с числами, то необходимо преобразовать идентификаторы (классы) в численное представление. Для этой задачи используется LabelEncoder, [9] который

кодирует имеющиеся классы в числовой аналог от 0 до n_classess-1, в случае рассматриваемого датасета, получится 10 классов от 0 до 9.

В итоге имеется набор данных вида: набор мел-кепстральнных коэффициентов аудиозаписи – номер класса. Получившееся данные будут использоваться для обучения и тестирования моделей машинного обучения.

Также использовался датасет LibriSpeech ASR Corpus [5]. Этот датасет содержит около 100 часов аудиоинформации с частотой дискретизации 16кГц, 251 человека, говорящего на английском языке.

Данный датасет аналогичным способом преобразуется в набор для обучения. Однако сам процесс предобработки требует значительных временных затрат, поэтому рекомендуется после преобразования аудиозаписей в мел-кепстральные коэффициенты хранить результаты преобразований в базе данных или csv файле – для последующей быстрой обработки при добавлении новых аудиозаписей в исходный датасет. Аудиодорожки данного датасета записаны в полной тишине, поэтому применение фильтров не требуется.

2.2. ВЫЧИСЛЕНИЕ MFCC

Располагая набором аудиоданных голосов, необходимо выделить сигнатуру этих голосов, используя MFCC коэффициенты. Для этого можно воспользоваться библиотеками librosa и numpy.

Первая библиотека служит для анализа аудиоданных. Вторая же необходима для работы с массивами, полученными из аудиоданных.

Из полученных на предыдущем этапе наборов аудиофайлов поочередно загружаем файлы в память для анализа. Для этого указываем путь к аудиофайлам и частоту дискретизации. Первоначальная аудиодорожка имела частоту дискретизации 44кГц, следовательно, ее части имеют ту же частоту. Используя метод load пакета librosa загружается аудиофайл в память как временной ряд с плавающей запятой.

Следующим этапом после загрузки является получение MFCC коэффициентов. В пакете librosa имеется метод feature.mfcc для получения этого набора коэффициентов со следующими параметрами:

- audio – загруженный в оперативную память аудиофайл
- sr – частота дискретизации аудиофайла (по умолчанию равна 22050 Гц)
- n_mfcc – параметр указывает, сколько извлечь коэффициентов аудиоряда для анализа. Обычно указывают от 20 до 40 коэффициентов, в случае если взять меньше, то велика вероятность потери данных, в следствии чего обработка голоса человека может быть неточной.
- n_fft – т.к. анализ временного ряда происходит не целиком, а окнами этот параметр задает размер этого окна. Чем больше окно, тем более точные данные можно получить, при этом расчет будет дольше.

После вычислений всех наборов мел-кепстральных коэффициентов, полученные массивы одной длины агрегируем по всем вычисленным участкам аудиофайла для нахождения мел-кепстральных коэффициентов всего временного ряда аудиодорожки [15].

Из полученного набора значений можно отбросить первые два значения, т.к. они покрывают мощность низких частот, которые человек не может воспроизвести.

В итоге имеется готовый набор значений для анализа голоса человека, при необходимости можно провести нормирование относительно максимального значения из этого набора.

2.3. РАЗДЕЛЕНИЕ ГОЛОСОВ НА ОСНОВЕ ОЦЕНКИ БЛИЗОСТИ ВЕКТОРОВ MFCC

Имея готовый набор коэффициентов, можно сравнить их с другими наборами, если таковые имеются.

```
test = os.listdir(dir_name)
test.sort()
for item in test:
    chunk = process_audio(dir_name+'/'+item)
    if not listvoice:
        listvoice.append(chunk)
```

```
        listfiles[dir_name+'/'+item]=0
        continue
    for idx, val in enumerate(listvoice):
        r=confidence(val, chunk)
        if r>d:
            end=True
            continue
        else:
            end=False
            listfiles[dir_name+'/'+item]=idx
            break;
    if end:
        listvoice.append(chunk)
        listfiles[dir_name+'/'+item]=len(listvoice)-1
```

Листинг 2.3.1 Алгоритм разделения аудио фрагментов

В коде (Листинг 2.3.1) происходит распределение аудиодорожек по голоса. Для этого из папки dir_name считываются все аудиодорожки, полученные на этапе подготовки данных, и ранжируются, чтобы не нарушить порядок диалога.

После этого для каждой аудиозаписи происходит расчет набора мел-кепстральных коэффициентов через метод process_audio, описанный на втором этапе. Получившейся набор сравнивается с уже вычисленными, если таковые отсутствуют, то он добавляется в список голосов для последующего использования. Далее идет расчёт следующей аудиодорожки.

В случае, если список голосов не пустой, для каждого голоса происходит расчет евклидового расстояния и выполняется сравнение с минимальным порогом различия голосов. Значение данного порога для пробы было решено установить равным 0,8. Если находится близкий по коэффициентам набор, то анализируемая аудиозапись добавляется в список с указанием номера голоса.

Если среди всех наборов коэффициентов голосов в списке не находится близкий относительно порога, то имеющийся набор мел-кепстральных коэффициентов добавляется в список как новый голос.

Полный листинг программной реализации метода разделения голосов относительно близости мел-кепстральных коэффициентов через расчет

евклидового расстояния на языке программирования Python представлена в приложении 1

2.4. РАЗДЕЛЕНИЕ ГОЛОСОВ НА ОСНОВЕ ПРИМЕНЕНИЯ МАШИННОГО ОБУЧЕНИЯ

Модуль классификации подразумевает использование методов машинного обучения для нахождения не разницы между голосами, а определение голоса определенного человека (принадлежности к определенному классу), тем самым это поможет однозначно идентифицировать говорящего по аудиодорожке.

Аудиодорожки преобразуются в наборы мел-кепстральных коэффициентов (MFCC). При помощи метода get_features, где на вход передается путь к аудиодорожке, вычисляются MFCC (Листинг 2.4.1).

Параметры для получения коэффициентов берутся с учетом исходных данных и эмпирическим путем, в данном случае из аудиоданных берется частота дискретизации, равная 16000 Гц, размер кадра для быстрого преобразования Фурье берется равным 4096 элементов для точности получаемого спектра, однако это влияет на длительность обработки и количество получаемых мел-кепстральных коэффициентов варьируется обычно от 20 до 40 значений, для точности, в данном случае берется набор, состоящий из 34 значений.

Из полученного набора можно вычленить первые два значения, т.к. они представляют собой мощность на самых низких частотах. Полученные данные являются признаками для обучения модели.

```
    def get_features(aname):
        audio, _ = librosa.load(aname, sr=44100)
        mfccs = librosa.feature.mfcc(y=audio, sr=44100,
n_mfcc=34,n_fft=4096)
        mfcc = mfccs / np.max(np.abs(mfccs))
        mfccs_processed = np.mean(mfcc.T,axis=0)
        return  mfccs_processed
```

Листинг 2.4.1 Метод get_features()

В работе используются две модели. Первая представляет собой 5-слойную нейронную сеть с двумя

полносвязными слоями, в каждом из которых 256 нейронов, функция активации ReLu. После каждого из двух полносвязных слоев идет слой dropout со значением 0.5. Выходной полносвязный слой с функцией активации softmax, с количеством нейронов, равных количеству классов в обучаемом датасете. Для датасета Surfingtech количество нейронов выходного слоя будет равно 10. (табл. 2.4.1).

Таблица 2.4.1

Модель с dropout

Слой	Количество нейронов	Количество параметров
Полносвязный слой (функция активации relu)	256	8960
Слой droupout (значение 0.5)	256	0
Полносвязный слой (функция активации relu)	256	65792
Слой droupout (значение 0.5)	256	0
Полносвязный слой (функция активации softmax)	10	2570
Итоговое количество параметров: 77 322		

Вторая модель является трехслойной и состоит только из полносвязных слоев, не включая droupout. Аналогично первой модели, два полносвязных слоя использую функцию активации ReLu, но имеют меньше количество нейронов, равных 64. Выходной слой полностью идентичен первой модели, при этом имеет меньшее количество параметров, как и вся модель в целом. (табл. 2.4.2).

Таблица 2.4.2

Модель без dropout

Слой	Количество нейронов	Количество параметров
Полносвязный слой (функция активации relu)	64	2240

Продолжение таблицы 2.4.2

Полносвязный слой (функция активации relu)	64	4160
Полносвязный слой (функция активации softmax)	10	650
Итоговое количество параметров: 7050		

Данные модели используют функцию categorial_crossentropy в качестве функции потерь. В качестве оптимизатора модели используется функция adam [11] с параметрами, представленными в таблице (табл. 2.4.3), так как она является самой эффективной среди других алгоритмов оптимизации в обучении нейронных сетей. Метрика, используемая для оценки работы моделей – accuracy.

Таблица 2.4.3

Параметры вызова функции Adam

Параметр	Значение
Adam_amsgrad	false
Adam_beta_1	0.9
Adam_beta_2	0.999
Adam_decay	0.0
Adam_epsilon	1.0E-7
Adam_learning_rate	0.001
curr_epoch	99
curr_step	9700
epochs	100
steps	97
Optimizer	Adam

Листинг программы с использованием методов машинного обучения представлен в приложении 2

2.5. ПЕРЕВОД РЕЧИ В ТЕКСТ

Для того чтобы преобразовать голос в текст используется библиотека SpeechRecognition, которая позволяет распознать речь при помощи разных

сервисов, таких как Google или IBM, а также является наиболее простой в использовании из всех других библиотек.

Работа с файлами происходит по средствам создание экземпляр класса Recognizer в котором имеется множество настроек и функций для распознавания речи из источника звука. У данного класса имеется 7 методов распознования речи из источника звука с использованием различных API:

- recognize_bing(): Microsoft Bing Speech
- recognize_google(): API Google Web Speech
- recognize_google_cloud(): Google Cloud Speech
- recognize_houndify(): Houndify от SoundHound
- recognize_ibm(): IBM Speech to Text
- recognize_sphinx(): CMU Sphinx
- recognize_wit(): wit.ai

Все методы кроме recognize_sphinx требуют для работы подключение к интернету. Для использования CMU Sphinx необходима установка дополнительного пакета PocketSphinx.

Для расшифровки речи в работе используется метод recognize_google, который использует ресурсы Google для распознания речи.

```
import speech_recognition as speech_recog
recog = speech_recog.Recognizer()
for path, voice in listfiles.items():
    sample_audio = speech_recog.AudioFile(path)
    with sample_audio as audio_file:
        audio_content = recog.record(audio_file)
    print('[голос'+str(voice)+']---')
    print(recog.recognize_google(audio_content,language="ru-
RU"))
```

Листинг 2.5.1 Алгоритм преобразования речи в текст

В коде (Листинг 2.5.1) для каждого элемента словаря содержащий номер голоса и путь к аудио фрагменту, считывается содержимое указанного аудио фрагмента, сохраняя данные в экземпляре класса AudioFile. После этого полученные аудио данные используются для создания экземпляра класса AudioData через метод record. После этих действий выводится информация о номере голоса и вызывается метод recognize_google,

преобразующий аудиоданные в текстовую информацию. На вход этому методу передается полученные данные в классе AudioData и указывается параметр language, который отвечает за язык перевода аудиоинформации. По умолчанию стоит перевод с английского языка, для русского языка указывается значение параметра равным «ru-RU» [12].

ГЛАВА 3. АНАЛИЗ ПОДХОДОВ К РАЗДЕЛЕНИЮ АУДИО СИГНАЛА

3.1. ТЕСТИРОВАНИЕ И ПОДБОР ПАРАМЕТРОВ

Первое тестирование программной реализации разделения голосов без использования машинного обучения проводилось на двух аудиофайлах: один с мужским и женским голосом, другой с двумя мужскими голосами.

Первый файл после предобработки разделяется ровно на два файла: мужской и женский голос.

Сравнение голосов выполняется путем вычисления расстояния между мел-кепстральными коэффициентами: чем оно меньше, тем больше вероятность, что это один и тот же голос.

В первом случае расстояние между мел-кепстральными массивами женского и мужского голосов равно 1.79111 условных единиц. Такое число можно считать достаточным для определения разности голосов.

Во втором случае разница между мел-кепстральными массивами двух мужских голосов получилась 2.15586. Такая разница в расстоянии обусловлена большим расстоянием между амплитудами частот.

При сравнении одного и того же голоса, записанного в разное время, евклидово расстояние равно 0.35368, такое значение можно считать минимальным для разности мел-кепстральных коэффициентов рассчитанных голосов. Однако, этот порог не является точным, при добавлении новых голосов потребуется новая оценка этого порога для определения различия между всеми имеющимися голосами в анализируемой аудиодорожке.

Далее тестирование проводилось на 15 секундном отрывке из фильма «Мстители», где происходит диалог между двумя мужчинами со следующим текстом:

- Видно, ты пришел воззвать к моей гуманности.
- Я вообще-то думал угрожать.
- Тогда стоило костюм надеть.
- Да… Он пообносился, а у тебя сияет жезл судьбы

После работы программы получился такой результат:

exporting chunk0.wav

exporting chunk1.wav

exporting chunk2.wav

exporting chunk3.wav

exporting chunk4.wav

exporting chunk5.wav

[голос0]---видно Ты пришёл воззвать моей гуманности

[голос1]---Я вообще-то думал угрожать

[голос0]---тогда тебе стоило костюмная

[голос2]---да

[голос1]---относился у тебя сессия

[голос3]---жезл судьбы

Весь диалог был разделен на 6 аудиодорожек с разным временным интервалом от 1 до 4 секунд.

На данном тесте были установлены следующие параметры:

- Минимальная длина тишины (min_silence_len) = 100 миллисекунд
- Порог тишины (silence_treshold) = -50 Дб (относительно полной шкалы)
- Порог разных голосов (d) = 0.8
- Частота дискретизации = 44100 Гц
- Количество мел-кепстарльных коэффициентов = 34
- Размер блока для быстрого преобразования Фурье = 2048

Как видно из приведенных результатов, количество голосов стало больше: в данном случае программа показывает, что их 4. Первые 3 аудиодорожки программа идентифицировала верно, хотя перевод 3 дорожки оказался не столь удачным. Оставшиеся аудиозаписи следовало отнести к голосу №1, однако короткие записи не позволили с высокой точностью идентифицировать их как «голос1».

Изменение параметров min_silence_len и silence_treshold не улучшило результат: либо количество записей увеличивалось во много раз и присутствовали пустые записи без слов, либо не происходило разделение на голоса и на выходе получалась изначальная аудиодорожка.

Однако увеличение размера блока до 4096 значений для быстрого преобразования Фурье немного улучшило результат работы алгоритма.

exporting chunk0.wav

exporting chunk1.wav

exporting chunk2.wav

exporting chunk3.wav

exporting chunk4.wav

exporting chunk5.wav

[голос0]---видно Ты пришёл воззвать моей гуманности

[голос1]---Я вообще-то думал угрожать

[голос0]---тогда тебе стоило костюмная

[голос2]---да

[голос1]---относился у тебя сессия

[голос0]---жезл судьбы

В данном случае последняя аудиодорожка была идентифицирована верно, суммарное количество голосов уменьшилось до 3.

Увеличение размера блока до 8192 значений не изменило результата. Также регулировались значения минимального порога различия голосов и количество получаемых мел-кепстральных коэффициентов. К увеличению точности результата это не привело.

Следующий пример был взят из кинофильма «Форсаж 3» с представленным диалогом:

- Для вас самое главное у кого побольше агрегат
- Я парень, у меня тачки в днк заложены. Так вот вы на чем гоняете, ничего игрушки
- Я тебя без тапочек ни сразу и узнала

В результате получилось следующее разбиение:

exporting chunk0.wav

exporting chunk1.wav

exporting chunk2.wav

exporting chunk3.wav

[голос0]---для вас самое главное у кого побольше агрегат

[голос1]---Я парень у меня тачки в ДНК заложено так вот вы на чём гоняете ничего игрушки

[голос0]---я тебя без тапочек Не сразу его

Хоть и перевод получился не самым удачным, но разделение по голосам дало точный результат. Однако стоит отметить, что в диалоге участвовали мужской и женский голоса, и, как правило, они достаточно различаются между собой. Параметры указаны были те же, что и на предыдущем тесте.

3.2. ОЦЕНКА КАЧЕСТВА МОДЕЛЕЙ

Обучение происходило на 100 эпохах, которые были сочтены достаточными для обучения моделей.

Изначально обучение происходило на датасете из 10 классов. Исходя из графиков точности и потерь на обеих моделях, они достаточно хорошо обучились и имели высокую скорость обучения. Представленные графики были получены из системы Comet ML.

Модель с droupout (рис.3.2.1) достигала точности в 90 % после 20 эпох обучения, а изменения потерь (рис. 3.2.2) после 30 эпох было незначительным.

Обучающая выборка 98.80%

Тестовая выборка 97.14%

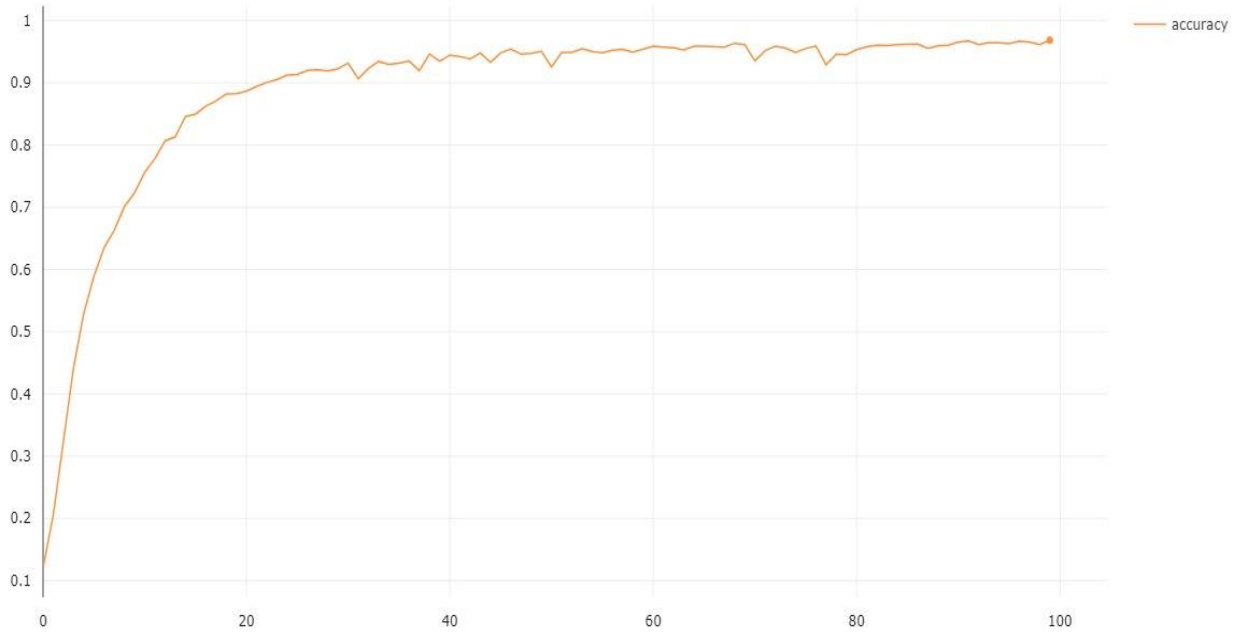


Рис.3.2.1. График точности для модели с dropout (Surfingtech)

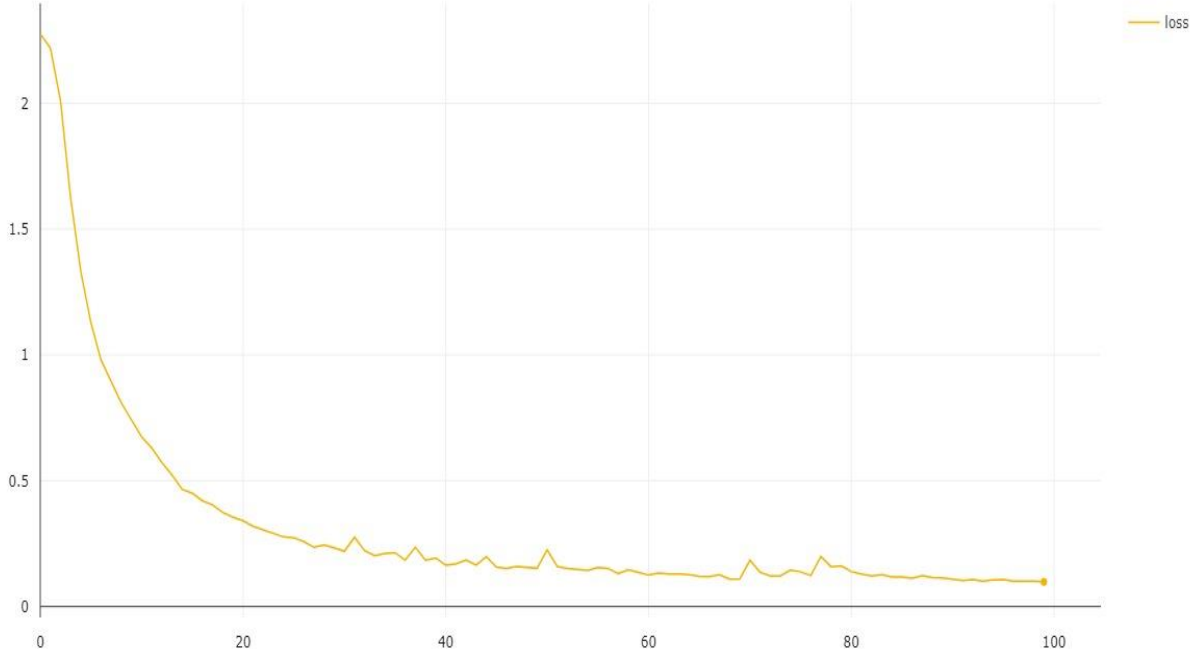


Рис. 3.2.2. График потерь для модели с dropout (Surfingtech)

Обучение модели без droupout имело чуть меньшую точность обучения и достигало 90% после 40 эпох обучения (рис. 3.2.3), потери плавно падали также после 40 эпох (рис. 3.2.4.).

Обучающая выборка 94.92%

Тестовая выборка 93.76%

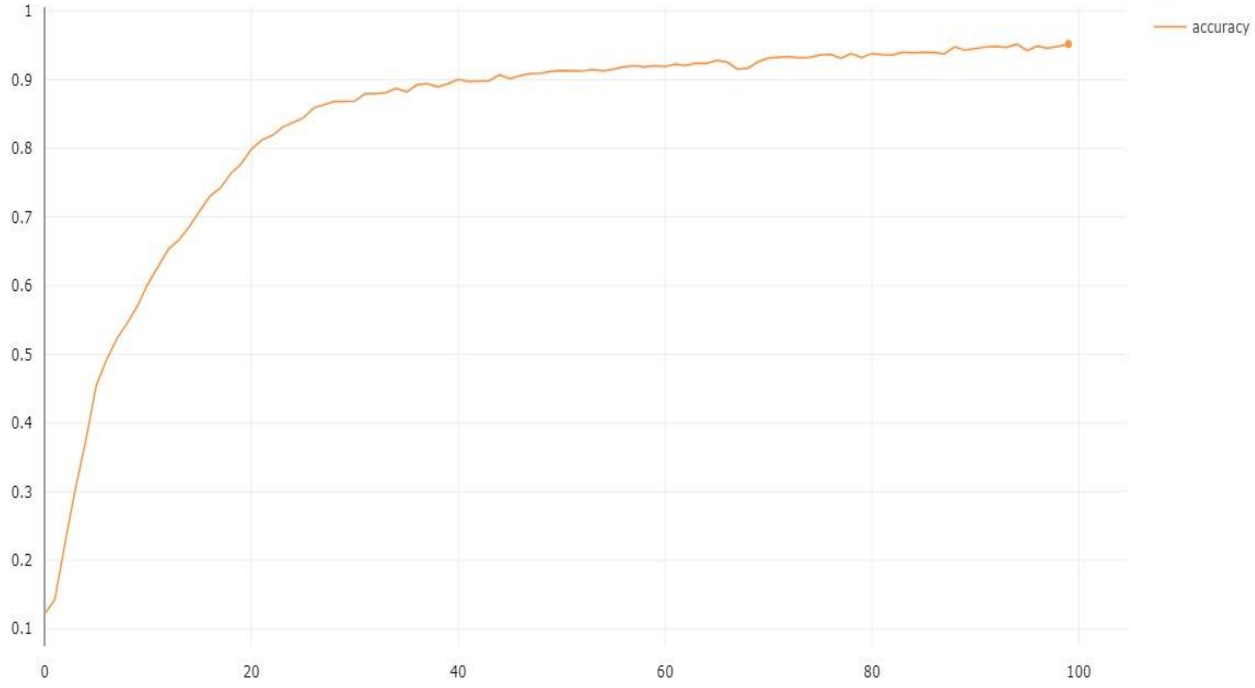


Рис. 3.2.3. График точности для модели без dropout (Surfingtech)

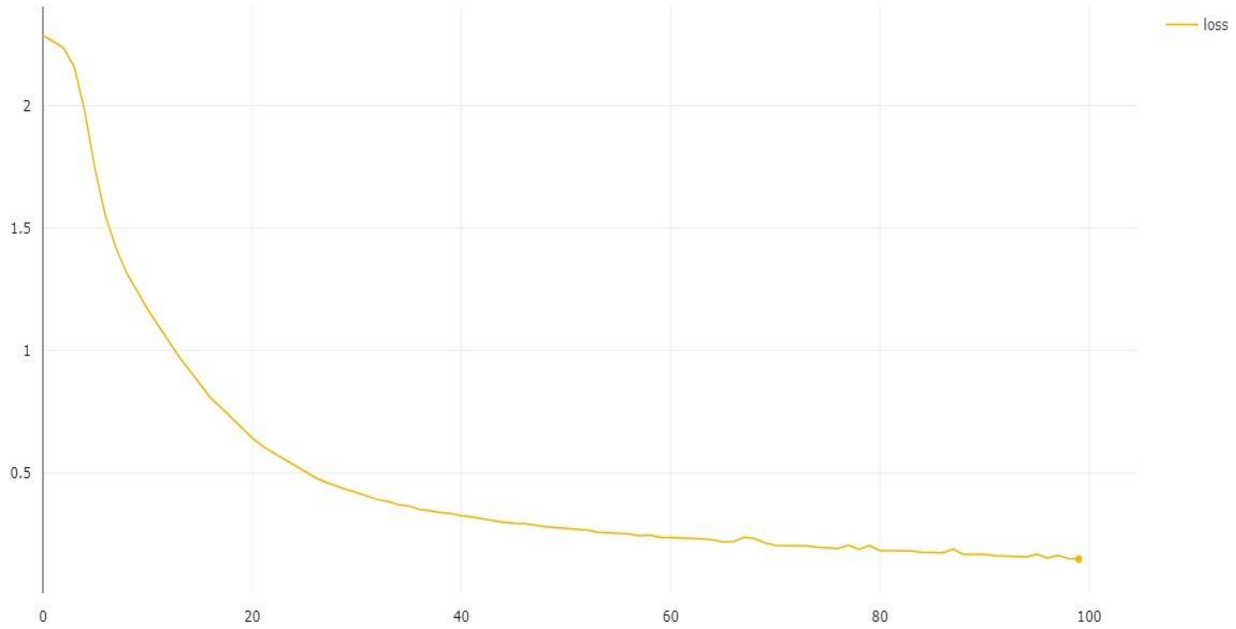


Рис. 3.2.4. График потерь для модели без dropout (Surfingtech)

Для следующего обучения использовался датасет из 26 129 аудиофайлов и 251 класса, предобработка такого датасета заняла около 4 часов, однако на этом датасете модели показали также высокую скорость

обучения и немного выше точность по сравнению с предыдущим датасетом. Общее количество обучаемых параметров изменилось для обеих моделей из-за повышения количества нейронов на последнем слое до 251 в соответствии с количеством классов. Для модели с dropout количество обучаемых параметров составило 139 259, для другой модели – 22 715.

Модель без droupout (рис. 3.2.5-3.2.6) достигала точности в 90 % после 50 эпох обучения, а изменения потерь после 50 эпох плавно стремились к нулю.

Обучающая выборка 98.09%

Тестовая выборка 95.22%

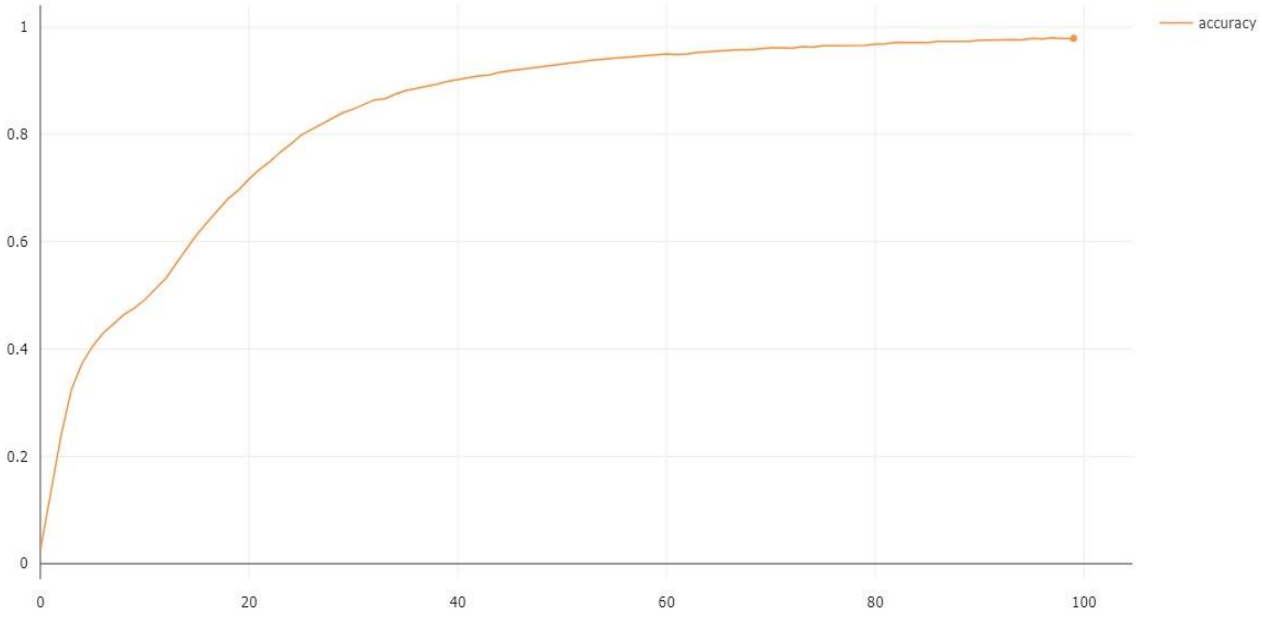


Рис. 3.2.5. График точности для модели без dropout (LibriSpeech)

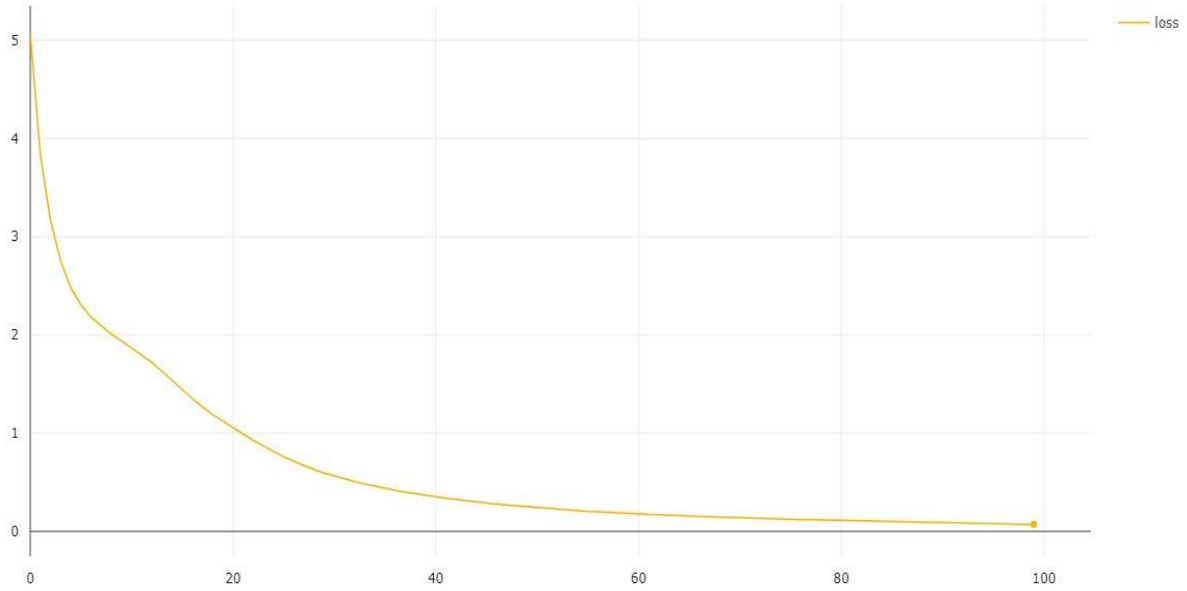


Рис.3.2.6. График потери для модели без dropout (LibriSpeech)

Модель с droupout (рис. 3.2.7-3.2.8) достигала точности в 90 % после 50 эпох обучения, а изменения потерь после 60 эпох были незначительными.

Обучающая выборка 99.54%

Тестовая выборка 99.12%

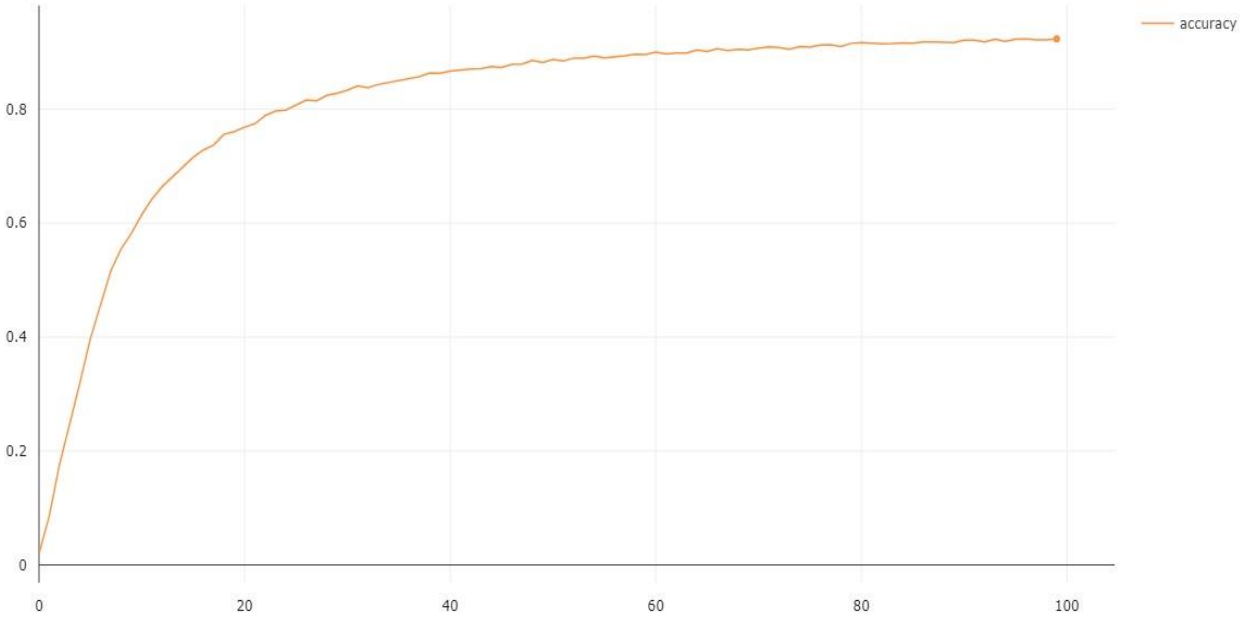


Рис.3.2.7. График точности для модели с dropout (LibriSpeech)

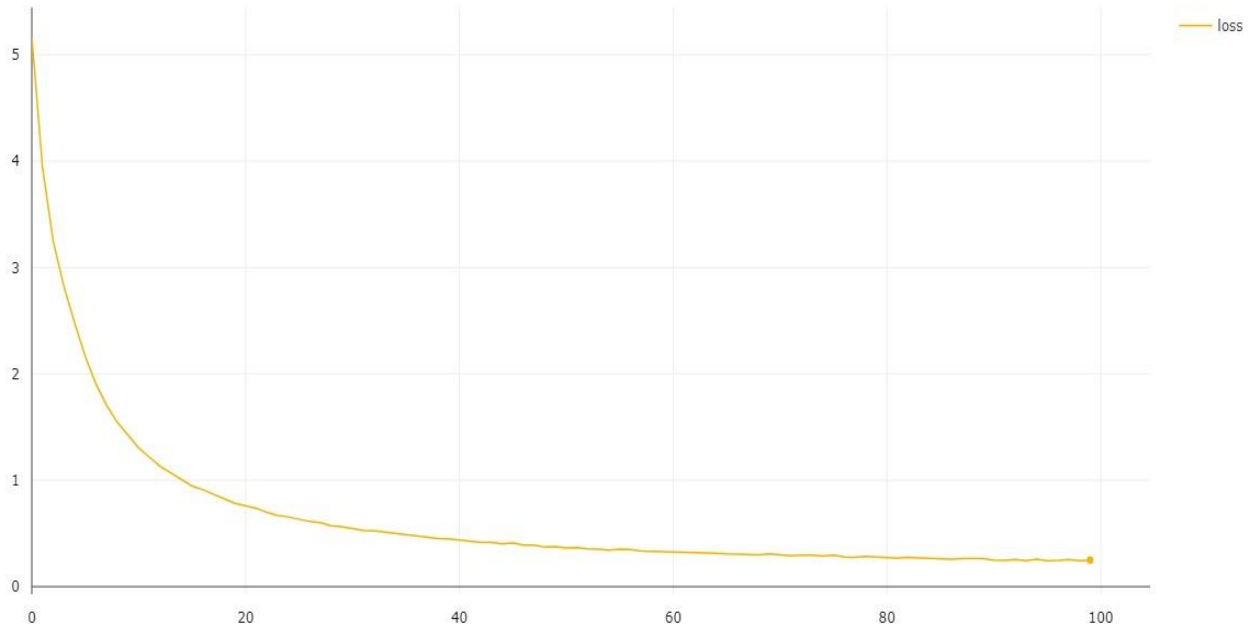


Рис.3.2.8. График точности для модели с dropout (LibriSpeech)

Имея обученные модели, можно с лёгкостью классифицировать новые аудиодорожки с человеческой речью.

В качестве примера возьмем аудиодорожку из датасета Surfingtech, со следующим текстом: «And they go through these amazing mental gymnastics to accomplish this».

Используя метод predict, посмотрим все вероятности принадлежности к каждому классу (табл. 3.2.1).

Таблица 3.2.1

Таблица вероятностей

Вероятность (в %)	Имя класса
65	F0001
0.03	F0002
0.003	F0003
34.82	F0004
0.004	F0005
0	M0001
0.122264	M0002
0.000011	M0003

Продолжение таблицы 3.2.1

0.000001	M0004
0.001055	M0005

Как видно из таблицы, наибольшая вероятность равна 65%, следовательно анализируемая запись принадлежит к классу f0001. Используемая аудио дорожка также принадлежит к классу f0001, следовательно предсказание является верным.

Что касается перевода, то используя тот же инструмент SpeechRecognition и сервисы google для преобразования речи в текст, имеем следующий результат: «[Голос f0001]---and they go through these amazing mental gymnastics to accomplish this».

3.3. СРАВНЕНИЕ ПОДХОДОВ К РАЗДЕЛЕНИЮ АУДИО СИГНАЛА

Преимуществом реализации без использования машинного обучения является то, что требуется только аудиодорожка для обработки, нет необходимости в обучении датасета.

Однако присутствуют недостатки:

- Возможно определение различия между голосами, но не их носителей.
- Требуется корректировать значения параметров для улучшения точности результатов.

Если смотреть со стороны конфиденциальности данных, то первый недостаток окажется преимуществом, так требуется сокрытие данных от пользователей или не требуется определять конкретную личность озвучивающую ту или иную аудиодорожку.

Второй недостаток крайне негативен, так как не позволит автоматизировать обработку аудиодорожек, и потребуется постоянное вмешательство стороннего специалиста для регулирования используемых параметров.

Однако даже если настроить параметры для определенного набора голосов, включая порог различия между этими голосами, то не факт что при добавлении нового голоса, эти параметры будут подходить для получения точных результатов.

Преимуществом подхода с использованием методов машинного обучения в представленной программной реализации являются следующие возможности:

- Идентификация каждого пользователя по его голосу с высокой точностью.
- Нет необходимости регулировать какие-либо параметры, такие как количество мел-кепстральных коэффициентов или размер обрабатываемого блока в быстром преобразовании Фурье. Для достаточно точных результатов берется размер кадра равным в 4096 элементов сигнала.
- Исключается параметр порога разности голосов, так как внутри модели машинного обучения происходит расчет вероятности принадлежности, получаемой на входе аудиодорожки, для каждого класса, на выход берется класс с наибольшей предсказанной точностью.

Однако у данного подхода есть недостаток: при добавлении нового голоса в набор данных, требуется заново переобучить модель и увеличить количество нейронов на выходе модели в соответствии с количеством имеющихся классов.

Так как обученная модель не имеет представления о новом классе, предсказание новой речи будет основываться на имеющихся классах в обученной модели. А также для работы данного подхода обязательно требуется размеченный набор данных.

Также подход с использование машинного обучения имеет возможность автоматизации всего процесса, как добавление новых голосов в систему, переобучения используемой модели и вывод предсказаний приходящим на вход аудиодорожкам.

Следовательно, для поддержания системы в рабочем состоянии не требуется постоянное наблюдение и изменение в реализации со стороны технического специалиста.

ЗАКЛЮЧЕНИЕ

В результате выполнения выпускной квалификационной работы были изучены специфика работы с аудиосигналом, использование преобразования Фурье для получения спектрограммы звукового сигнала с последующим получение из спектра мел-кепстральных коэффициентов.

Были описаны и реализованы методики разделения аудиодорожки на фрагменты относительно участков тишины с сохранением порядка следования, получения спектрограммы и мел-кепстральных коэффициентов, а также определена технология для преобразования речи человека в текстовый аналог.

Были реализованы следующие подходы для персонализации речи:

- Первый подход использовал предобработку аудиоданных для получения MFCC и используя евклидово расстояние сравнивал между с собой полученные коэффициенты на различие голосов с последующим переводом их в текст. Для тестирования данного подхода из аудиорядов фильмов были вырезаны и записаны специальные аудиодорожки.
- Второй подход использовал методы машинного обучения для персонификации говорящего человека и для проверки работоспособности данного подхода были найдены два англоязычных датасета.

Также был проведен анализ реализованных подходов к персонализации личностей. Расчеты показали, что подход, использующий евклидово расстояние, оказался крайне неэффективным с точки зрения использования из-за его недостатков в плане регулирования параметров обработки аудиоданных. Второй подход по реализации обработки аудиодорожек оказался более успешным и в плане использования лучше подходит для решения задачи.

В качестве дальнейшей перспективы по улучшению персонификации аудиодорожек человеческой речи можно использовать аудиофильтры или

использовать решение от компании Deezer под названием Spleeter. В описании сказано, что данная система используется для разделения вокала и музыкальных инструментов, но также применима аудиодорожкам из фильмов, для получения только речи человека. В целом это можно применить не только к фильмам, но и к другим возможным ситуациям.

Реализованная технология может быть применена в качестве модуля во множестве различных сфер от видео на YouTube, где для каждого голоса будет персонификация говорящего в субтитрах для видео или для структуризации озвученного текста в онлайн курсах или в видеоуроках.

СПИСОК ЛИТЕРАТУРЫ

1. Comet – Build better models faster! // Comet: [сайт]. URL: https://www.comet.ml/site/ (дата обращения 02.06.2021)
2. Deep Learning library for Theano and Tensorflow Keras documentation // Keras: [сайт]. URL: https://faroit.com/keras-docs/1.2.0/ (дата обращения 25.05.2021)
3. Free ST American English Corpus DataSet // OpenSLR: [сайт]. URL: http://www.openslr.org/45/ (дата обращения 23.05.2021)
4. Keras – Comet ML // Comet: [сайт]. URL: https://www.comet.ml/docs/python-sdk/keras/ (дата обращения 02.06.2021)
5. LibriSpeech ASR Corpus DataSet // OpenSLR: [сайт]. URL: http://www.openslr.org/12/ (дата обращения 23.05.2021)
6. Librosa 0.8.1 documentation // Librosa: [сайт]. URL: https://librosa.org/doc/latest/index.html (дата обращения 25.05.2021)
7. Mark Heath, NAudio is an open source .NET audio library [сайт]. URL: https://github.com/naudio/NAudio (Дата обращения: 02.06.2021).
8. Pydub // PyPi: [сайт]. URL: https://pypi.org/project/pydub/ (дата обращения 25.05.2021)
9. Sklearn.preprocessing.LabelEncoder // SkLearn: [сайт]. URL: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html (дата обращения 23.05.2021)
10. SpeechRecognition // PyPi: [сайт]. URL: https://pypi.org/project/SpeechRecognition/ (дата обращения 25.05.2021)
11. Tf.keras.optimizers.Adam // TensorFlow: [сайт]. URL: https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adam (дата обращения 25.05.2021)
12. The ultimate guide to speech recognition with Python // RealPython: [сайт], - https://realpython.com/python-speech-recognition/ (Дата обращения: 02.06.2021).

13. Window function // Wikipedia: [сайт]. URL: https://en.wikipedia.org/wiki/Window_function#Triangular_window (дата обращения 25.05.2021)
14. Алюнов Д.Ю., Сергеев Е.С., Пигачев П.В., Мытников А.Н. Реализация алгоритма обработки и распознавания речи // Современные наукоемкие технологии. Москва, 2016. – № 3-2. – С. 225-230; URL: http://www.top-technologies.ru/ru/article/view?id=35724 (дата обращения: 05.06.2021).
15. Анализ аудиоданных с помощью глубокого обучения и Python (Часть 1) // Nuances of Programming: [сайт]. URL: https://nuancesprog.ru/p/6713/ (дата обращения 23.05.2021)
16. Заковряшин А., Малинин П., Лепендин А Применение распределений мел-частотных кепстральных коэффициентов для голосовой идентификации личности // Известия Алтайского государственного университет. Барнаул, 2014. – № 1. С. 156 – 160; URL: https://cyberleninka.ru/article/n/primenenie-raspredeleniy-mel-chastotnyh-kepstralnyh-koeffitsientov-dlya-golosovoy-identifikatsii-lichnosti (Дата обращения: 02.06.2021).
17. Как работает нейронная сеть: алгоритмы, обучение, функции активации и потери // Neurohive: [сайт]. URL:https://neurohive.io/ru/osnovy-data-science/osnovy-nejronnyh-setej-algoritmy-obuchenie-funkcii-aktivacii-i-poteri/ (Дата обращения: 02.06.2021).
18. Ласкарис Нико, Как применить методы машинного обучения и глубокого обучения к аудио анализу [сайт]. URL: https://www.machinelearningmastery.ru/how-to-apply-machine-learning-and-deep-learning-methods-to-audio-analysis-615e286fcbbc/ (Дата обращения: 02.06.2021).
19. Лукин А. Спектроанализатор – что мы на нем видим? [сайт]. URL: https://prosound.ixbt.com/education/spektr-analys.shtml (Дата обращения: 02.06.2021).

20. Маркина Ю.Ю., Белов Ю.С. Кепстральные коэффициенты как необходимая характеристика процесса создания системы имитации голоса человека с помощью методов глубокого обучения // Международный студенческий научный вестник. Москва, 2018. – № 1.; URL: http://www.eduherald.ru/ru/article/view?id=18125 (дата обращения: 31.05.2021).
21. Мел-кепстральные коэффициенты (MFCC) и распознавание речи // Habr: [сайт]. URL: https://habr.com/ru/post/140828/ (Дата обращения: 02.06.2021).
22. Методы оптимизации нейронных сетей // Habr: [сайт]. URL: https://habr.com/ru/post/318970/ (Дата обращения: 02.06.2021).
23. Нейронная сеть (Neural network) // Loginom: [сайт]. URL: https://wiki.loginom.ru/articles/neural-network.html (Дата обращения: 02.06.2021).
24. Нейронная сеть // Википедия: [сайт]. URL: https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C (Дата обращения: 02.06.2021).
25. Преобразование Фурье в действии: точное определение частоты сигнала и выделение нот // Habr: [сайт]. URL: https://habr.com/ru/post/196374/ (Дата обращения: 02.06.2021).
26. Простыми словами о преобразовании Фурье // Habr: [сайт], - https://habr.com/ru/post/196374/ (Дата обращения: 02.06.2021).
27. Распознавание речи для чайников // Habr: [сайт]. URL: https://habr.com/ru/post/226143/ (Дата обращения: 02.06.2021).
28. Текстонезависимая идентификация по голосу // Habr: [сайт]. URL: https://habr.com/ru/post/336516/ (Дата обращения: 02.06.2021).
29. Теорема Найквиста-Шеннона-Котельникова // Digital Music Academy: [сайт]. URL: https://digitalmusicacademy.ru/lesson-nyquist-theorem (дата обращения 25.05.2021)

30. Функции активации нейросети: сигмоида, линейная, ступенчатая, ReLu, than // Neurohive: [сайт]. URL:https://neurohive.io/ru/osnovy-data-science/activation-functions/ (Дата обращения: 02.06.2021).

ПРИЛОЖЕНИЕ

Приложение 1

Программная реализация разделения голосов с использованием евклидовой меры близости

```
from pydub import AudioSegment
from pydub.silence import split_on_silence
import os
import shutil
shutil.rmtree('recordings')

#путь к исходному файлу
sound_file = AudioSegment.from_wav("audio6.wav")
#Разделение на аудиодорожки
audio_chunks = split_on_silence(sound_file,
min_silence_len=150, silence_thresh=-46 )

os.mkdir('recordings')
#Сохранение полученных аудиодорожек
for i, chunk in enumerate(audio_chunks):
   out_file = "chunk{0}.wav".format(i)
   print("exporting", out_file)
   chunk.export('recordings/'+out_file, format="wav")

import librosa as lr
import numpy as np

#Метод получения мел-кепстральных коэффициентов
def process_audio(aname):
  audio, _ = lr.load(aname, sr=SR) # Загружаем трек в память
  # Извлекаем коэффициенты
  mfcc = lr.feature.mfcc(audio,
                         sr=SR,
                         n_mfcc=34,
                         n_fft=4096)
  # Суммируем все коэффициенты по времени
  # Отбрасываем два первых, так как они не слышны человеку и
содержат шум
  mfcc = np.sum(mfcc[2:], axis=-1)
  # Нормализуем их
  mfcc = mfcc / np.max(np.abs(mfcc))
  return mfcc

#Вычисление евклиодового расстояния
def confidence(x, y):
  return np.sum((x - y)**2)

d=0.8 # Порог разницы голосов
SR = 44000 # Частота дискретизации
listvoice = []
listfiles = {}
```

```
    end = False
    #Загружаем аудиодорожки
    dir_name = "recordings"
    test = os.listdir(dir_name)
    test.sort()
    #Вычилсение сигнатуры аудиодорожки голоса и сравнение с
порогом
    #На выходе список вида: (голос - путь к аудиодорожке)
    for item in test:
        chunk = process_audio(dir_name+'/'+item)
        if not listvoice:
            listvoice.append(chunk)
            listfiles[dir_name+'/'+item]=0
            continue
        for idx, val in enumerate(listvoice):
            r=confidence(val, chunk)
            if r>d:
                end=True
                continue
            else:
                end=False
                listfiles[dir_name+'/'+item]=idx
                break;
        if end:
            listvoice.append(chunk)
            listfiles[dir_name+'/'+item]=len(listvoice)-1

    #Преобразование размеченных аудиодорожек в текст
    import speech_recognition as speech_recog
    recog = speech_recog.Recognizer()
    for path, voice in listfiles.items():
        sample_audio = speech_recog.AudioFile(path)
        with sample_audio as audio_file:
            audio_content = recog.record(audio_file)
        print('[голос'+str(voice)+']---')
        try:

print(recog.recognize_google(audio_content,language="ru-RU"))
        except Exception:
            print()
```

Приложение 2

Программная реализация разделения голосов при помощи машинного обучения

```
from comet_ml import Experiment
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import Dense, Dropout,Activation
from keras.utils import to_categorical
import numpy as np
import pandas as pd
import os
import librosa
#Создание эксперимента для Comet ML
experiment = Experiment(
    api_key="********",
    project_name="vkr",
    workspace="isaac30",
)

#Получение мел-кепстральных коэффициентов
def get_features(aname):
    audio, _ = librosa.load(aname, sr=44100)
    mfccs    =    librosa.feature.mfcc(y=audio,    sr=44100,
n_mfcc=34,n_fft=4096)
    mfcc = mfccs / np.max(np.abs(mfccs))
    mfccs_processed = np.mean(mfcc.T,axis=0)
    return  mfccs_processed

datadir='C:/Users/User/.spyder-py3/dataset'
listsound = {}


#Чтение аудиофайлов
for file in os.listdir(datadir):
        listsound[datadir+'/'+file]=file[:file.find("_")]
#for r, d, f in os.walk(datadir):
    #for file in f:
        #if file.endswith(".flac"):
            #listsound[os.path.join(r,
file)]=file[:file.find("-")]

#Получение признаков и классов
myfeatures = []
for audio, label in listsound.items():
    class_name = label
    datamfcc = get_features(audio)
    myfeatures.append([datamfcc, class_name])
myfeatures            =            pd.DataFrame(myfeatures,
columns=['feature','class_name'])
#Преобразование классов для классификации
```

```
    X = np.array(myfeatures.feature.tolist())
    y = np.array(myfeatures.class_name.tolist())
    le = LabelEncoder()
    yy = to_categorical(le.fit_transform(y))

    #Разделение данных на обучающую и тестовую выборки
    x_train, x_test, y_train, y_test = train_test_split(X, yy,
test_size=0.2,stratify = yy)
    num_labels = yy.shape[1]

    def create_model(input_shape=(34,)):
        #Модель 1
        model = Sequential()
        model.add(Dense(64))
        model.add(Activation('relu'))
        model.add(Dense(64))
        model.add(Activation('relu'))
        model.add(Dense(num_labels))
        model.add(Activation('softmax'))
        #Модель 2
        #model.add(Dense(256))
        #model.add(Activation('relu'))
        #model.add(Dropout(0.5))
        #model.add(Dense(256))
        #model.add(Activation('relu'))
        #model.add(Dropout(0.5))
        #model.add(Dense(num_labels))
        #model.add(Activation('softmax'))
        model.compile(loss='categorical_crossentropy',
metrics=['accuracy'], optimizer='adam')

        return model
    model = build_model_graph()

    #Обучение и оценка моделей
    epochs = 100
    batch_size = 32
    model.fit(x_train,      y_train,      batch_size=batch_size,
epochs=epochs, verbose=1)
    print("Точность         на         обучающей         выборке:
{0:.2%}".format(model.evaluate(x_train, y_train, verbose=1)[1]))
    print("Точность         на         тестовой         выборке:
{0:.2%}".format(model.evaluate(x_test, y_test, verbose=1)[1]))
    model.summary()
    experiment.end()
```