



Министерство образования и науки Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технический университет  
имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)

---

ФАКУЛЬТЕТ “Информатика и системы управления”

КАФЕДРА “Программное обеспечение ЭВМ и информационные технологии”

---

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ  
НА ТЕМУ:**

Метод выделения голосовой составляющей из  
монофонического аудио сигнала

Студент ИУ7-82  
(Группа)

Танцевов Г.М.  
(Подпись, дата) (И.О.Фамилия)

Руководитель ВКР

Оленев А.А.  
(Подпись, дата) (И.О.Фамилия)

Консультант

(Подпись, дата) (И.О.Фамилия)

Консультант

(Подпись, дата) (И.О.Фамилия)

Нормоконтролер

(Подпись, дата) (И.О.Фамилия)

2018 г.

T3

T3

план

# Реферат

РПЗ 67 страниц, 20 рисунков, 11 таблиц, 21 источник.

Объектом исследования данной работы являются музыкальные произведения. Целью работы является разработка метода автоматизированного выделения голосовой составляющей из музыкальной композиции.

Задачи:

- анализ предметной области;
- анализ методов выделения вокальной партии из музыкальных композиций;
- разработка собственного метода на основании проанализированных;
- разработка программного комплекса, реализующего выбранный метод;
- анализ эффективности работы программного продукта.

Способы применения программного продукта: программный продукт может применяться для дальнейшей разработки выделения компонентов сведенного аудио сигнала, выделения голосовой составляющей из песен для автоматизированного создания текста, выделения голосовой составляющей из аудио дорожки видеофайла для улучшения автоматизированной генерации субтитров.

# Содержание

Введение . . . . .	11
1 Аналитический раздел . . . . .	13
1.1 Цифровое представление аудио сигналов . . . . .	14
1.2 Существующие методы выделения источников . . . . .	15
1.2.1 Метод главных компонент . . . . .	16
1.2.2 Факторизация неотрицательных матриц . . . . .	17
1.2.3 Адаптивный метод FASST . . . . .	19
1.2.4 Методы, использующие нейронные сети . . . . .	20
1.2.4.1 Общий принцип работы нейронной сетей . . . . .	20
1.2.4.2 Глубинные нейронные сети . . . . .	23
1.2.4.3 Сверточные нейронные сети . . . . .	25
1.2.4.4 Рекуррентные нейронные сети . . . . .	29
1.3 Оценка методов выделения источников . . . . .	30
1.4 Сравнение методов . . . . .	33
1.5 Вывод . . . . .	34
2 Конструкторский раздел . . . . .	36
2.1 Архитектура программного продукта . . . . .	36
2.2 Алгоритмы . . . . .	36
2.2.1 Обработка входных данных . . . . .	38
2.2.2 Генерация частотно-временных масок . . . . .	38
2.2.3 Описание сети . . . . .	38
2.2.3.1 Обучение нейронной сети . . . . .	40
2.2.4 Формирование выходных данных . . . . .	40
2.3 Используемые структуры данных . . . . .	40
2.4 Тестирование . . . . .	42
2.4.1 Тестовые данные . . . . .	42
2.5 Вывод . . . . .	43
3 Технологический раздел . . . . .	44
3.1 Выбор средств разработки . . . . .	44
3.1.1 Выбор языка программирования . . . . .	44
3.1.2 Выбор среды программирования и отладки . . . . .	44

3.1.3	Используемые библиотеки . . . . .	45
3.2	Система контроля версий . . . . .	45
3.3	Требования к вычислительной системе . . . . .	45
3.4	Формат данных . . . . .	46
3.5	Диаграмма классов . . . . .	46
3.5.1	Описание классов . . . . .	46
3.6	Построение нейронной сети . . . . .	50
3.7	Установка программного обеспечения . . . . .	54
3.8	Руководство пользователя . . . . .	54
3.8.1	Режим обучения . . . . .	54
3.8.2	Нормальный режим . . . . .	55
3.9	Вывод . . . . .	56
4	Экспериментальный раздел . . . . .	58
4.1	Описание используемой модели . . . . .	58
4.2	Описание тестирующей выборки . . . . .	58
4.3	Методика проведения исследования . . . . .	59
4.4	Спектрограммы . . . . .	60
	Заключение . . . . .	65
	Список использованных источников . . . . .	66

## Глоссарий

В настоящей работе используются следующие термины с соответствующими определениями.

**Дискретизация** — преобразование непрерывного информационного множества аналоговых сигналов в дискретное множество.

**Музыкальное произведение** — последовательность сочетаний звуков.

**Звук** — физическое явление, представляющее собой распространение в виде упругих волн механических колебаний в твёрдой, жидкой или газообразной среде.

**Колебания** — повторяющийся в той или иной степени во времени процесс изменения состояний системы около точки равновесия. Колебания характеризуются частотой, фазой и амплитудой.

**Частота** — физическая величина, характеристика периодического процесса, равна количеству повторений или возникновения событий в единицу времени. Единица измерения — герцы (Гц).

**Фаза** — аргумент периодической функции, описывающей колебательный или волновой процесс.

**Амплитуда** — максимальное значение смещения или изменения переменной величины от среднего значения при колебательном или волновом движении.

Применительно к музыке вместо общего термина «звук» используют более конкретный термин «музыкальный звук».

**Музыкальный звук** — это звук определённой высоты, использующийся как материал для создания музыкальных сочинений. Для письменной фиксации музыкальных звуков используется нотация. Наиболее распространённые формы записи высотных значений музыкальных звуков — латинская буквенная (C, D, E, F, G, A, B/H) или слоговая (до, ре, ми, фа, соль, ля, си) нотация. В данной работе для обозначения музыкальных звуков используется латинская буквенная нотация с указанием номера октавы.

**Нота** — музыкальный звук или его письменная запись.

Далее понятие «звук» используется в смысле музыкального звука.

**Аккомпанемент** — сопровождение одним или несколькими инструментами, а также оркестром сольной партии (певца, инструменталиста, хора и других). Сопроводителя называют аккомпаниатором. Аккомпанементом также называют гармоническое и ритмическое сопровождение основной мелодии, голоса.

**Вокал (или пение)** — исполнение мелодии с помощью голоса человека.

**Тембр** — (обертоновая) окраска звука, «качество тона».

**Рефакторинг** — процесс изменения внутренней структуры программы, не затрагивающий её внешнего поведения и имеющий целью облегчить понимание её работы.

**Цифровая звуковая рабочая станция** — электронная или компьютерная система, предназначенная для записи, хранения, редактирования и воспроизведения цифрового звука. Предусматривает возможность выполнения на ней законченного цикла работ, от первичной записи до получения готового результата.

## **Обозначения и сокращения**

**ГНС** — Глубинные нейронные сети

**СНС** — Сверточные нейронные сети

**РНС** — Рекуррентная нейронная сеть

**ОПФ** — Оконное преобразование Фурье

**ФНМ** — Факторизация неотрицательных матриц

**FASST** — Flexible Audio Source Separation Toolbox

## Введение

Информационные технологии в настоящее время имеют всё большее влияние в различных науках. Так за последние двадцать лет развитие получило такое направление как получение информации о музыке (англ. music information retrieval, MIR)[1]. Это направление включает в себя анализ музыкальных произведений и связанных с ними метаданных с помощью вычислительных методов и объединяет в себе идеи музыковедения, цифровой обработки сигналов и машинного обучения. К задачам MIR относятся определение жанра, транскрибирование, построение рекомендательной системы, определение настроения музыкального произведения, выделение источников, определение музыкальных инструментов и другие. На данный момент каждая задача является нетривиальной и не решена полностью[2].

Выделение источников является важной задачей, решение которой найдет применение во многих реальных задачах. Например, точность систем автоматизированного распознавания голоса (англ. automatic speech recognition, ASR) может быть увеличена с помощью выделения шума из аудио сигнала[3]. Точность распознавания аккордов и оценки высоты тона можно улучшить, отделив вокал от музыки[4]. Автоматизацию создания нотных партитур для музыкальных произведений можно провести с помощью методов автоматического транскрибирования к выделенным из исходного сигнала сигналы источников.

Методы выделения источников подразумевают определение и воспроизведение аудио сигналов, являющихся компонентами исходного сигнала.

Полное выделение источников из музыкального произведения включает в себя задачу классификации использованных при записи источников, их определение и воспроизведение. Но на данный момент решение задачи полного выделения источников из полифонического музыкального произведения считается крайне сложным или невозможным [5]. По этому целью систем автоматического выделения аудио источников обычно является выделение заранее известных источников, например вокал, бас, барабаны.

Целью данной работы является разработка метода выделения голосовой составляющей из монофонического аудио сигнала. Для решения поставленной задачи необходимо:

- проанализировать предметную область;
- проанализировать существующие методы выделения источников из аудио сигнала;
- на основе полученных во время анализа данных разработать собственный метод выделения голосовой составляющей из монофонического аудио сигнала;
- реализовать разработанный метод в программном продукте.

# 1 Аналитический раздел

Выделение голосовой составляющей из аудио сигнала является частным случаем задачи разделения комбинации аудио сигналов на составляющие. Музыкальное произведение может быть записано как с использованием нескольких микрофонов, захватывающих разные источники, при этом изоляция источников будет ограничена, либо с использованием выделенной на каждый источник. Все записанные сигналы в последствии проходят процесс сведения для получения итоговой аудио записи. Тем самым конечный аудио сигнал можно представить в виде суммы отдельных источников с применением фильтров к каждому источнику.

Математически это можно записать как:

$$x(t) = \sum_{j=1}^J \sum_{\tau=-\infty}^{\infty} a_j(t - \tau, \tau) s_j(\tau) \quad (1.1)$$

где

- $x$  — итоговый аудио сигнал;
- $s_j$  — сигнал источника;
- $J$  — количество источников;
- $a_j$  — фильтр, примененный к источнику в процессе сведения.

Если применяемые фильтры являются линейными, то итоговый сигнал представляет собой линейную комбинацию. С другой стороны, к итоговому сигналу так же могут быть применены фильтры. С этой точки зрения сведенный сигнал перестает быть линейной суммой отдельных источников.

Тем самым, итоговая запись, в контексте задачи выделения аудио источников, может быть определена по следующим признакам[6]:

- a) Количество используемых во время записи источников аудио сигнала. Источники могут быть записаны как отдельно, с помощью выделенных линий звукозаписи либо нескольких микрофонов, так и при помощи одного микрофона, захватывающего все звуки одновременно. С понижением количества звукозаписывающих устройств повышается смешанность отдельных аудио сигналов и влияние на них шумов.

б) Количество примененных к итоговому аудио сигналу эффектов. С учетом итоговых фильтров формула 1.1 принимает вид:

$$x(t) = \prod_{k=1}^K \psi_k(t) \sum_{j=1}^J \sum_{\tau=-\infty}^{\infty} a_j(t - \tau, \tau) s_j(\tau) \quad (1.2)$$

где  $\psi_k(t), k = 1\dots, K$  – применяемые к итоговому сигналу фильтры. Исходя из формулы, применение фильтров к итоговому сигналу усложняет задачу выделения аудио источников.

в) Изменение положения звукозаписывающих устройств во время записи. Микрофоны, ведущие запись, могут быть статичными, находясь в закрепленном состоянии, либо перемещаться, например от одного источника к другому. Второй случай оказывает непосредственное влияние на сигналы аудио источников, добавляя шумы и изменяя уровень сигнала.

Эта работа будет сфокусирована на обработке записей без итоговых фильтров, источники которых были записаны независимо друг от друга без изменений положения звукозаписывающих устройств.

## 1.1 Цифровое представление аудио сигналов

Звук имеет три основных характеристических параметров: амплитуда, частота и фаза. Две последние характеристики являются функциями времени, в то время как амплитуда определяет динамический диапазон. По этому в цифровом представлении аудио сигнала записываются изменения амплитуды как функция времени. Тем самым процесс отцифровки аналогового аудио сигнала является записью мгновенных амплитудных значений (дискретизация по амплитуде) в постоянные моменты времени (дискретизация по времени). Графическое представление дискретизации аналогового сигнала представлено на рисунке 1.1.

Для определения периода записи амплитудных значений задается частота дискретизации.

Существует теорема Котельникова[7], утверждающая, что «любую функцию  $F(t)$ , состоящую из частот от 0 до  $f_1$ , можно непрерывно передавать с любой точностью при помощи чисел, следующих друг за другом через  $1/(2f_1)$  секунд».

При максимальной воспринимаемой человеческим ухом частоте в 20 кГц, по теореме Котельникова минимальная необходимая частота дискретизации должна быть 40 кГц.

Стандартная частота дискретизации аудио сигнала составляет 44,1 кГц, максимальная – 192 кГц.

Для определения максимального значения амплитуды используется значение квантования (или разрядность), задающаяся в битах. В зависимости от используемого формата разрядность может быть 1, 8, 16, 24 и 32 бит.

В данной работе будут использоваться wav-файлы (сокращение Waveform Audio File Format). Формат WAVE был разработан корпорацией Майкрософт, является контейнером для хранения и записи оцифрованных аудиопотоков. Представляет из себя сжатый формат, отличается отсутствием потери качества и основан на расширении RIFF. Структура wav-файлов имеет чёткую форму, описанную в стандарте [8].

## 1.2 Существующие методы выделения источников

Методы выделения источников делятся на две основные категории[9].

— Слепое выделение. Методы, относящиеся к данной категории, обрабатывают смесь аудио сигналов без помощи, или с минимальным количеством, информации об источниках и методах их сведения. Наиболее популярные из этих алгоритмов используют (предполагаемую) статистическую независимость исходных сигналов. Примером являются Метод главных компонент [10] [11], Факторизация неотрицательных матриц. С ростом количества исследований в области нейронных сетей возникла идея применения методов глубинного обучения в задачах выделения источников. Работы по данной теме были сфокусированы на таких сетях как Глубинные и Сверточные нейронные сети.

— Информированное выделение. Для данной категории методов предполагается наличие предварительной информации об исходных сигналах, в форме MIDI, партитуры или другом виде представления музыки[12]. Целью данной работы не является изучение информированного выделения, потому детального описания представлено не будет.

Данная работа будет сфокусирована на методах слепого выделения.

### 1.2.1 Метод главных компонент

Метод главных компонент (англ. Principal Component Analysis) и анализ независимых компонент (англ. Independent Component Analysis) – это статистические способы уменьшения размерности данных, который являлся объектом для исследований в области выделения аудио источников [10] [11].

Основная идея этого метода заключается в том, чтобы проецировать данные из временных рядов, таких как аудиозапись, в новые системы отсчета, которые основаны на некотором статистическом критерии. Эти оси являются статистически независимые в отличие от преобразования Фурье, где данные временной области проецируются на оси частот, которые могут перекрываться. Частотные оси в преобразовании Фурье остаются неизменными независимо от анализируемой части, тогда как в методе главных компонент и анализе независимых компонент оси являются динамическими и различны для каждой анализируемой части. После нахождения, оси, на которые происходит проецирование, могут быть разделены и инвертированы для нахождения источников, представленных в исходном сигнале.

Формулу 1.1 можно представить в виде:

$$x(t) = As(t) \quad (1.3)$$

где

- $A$  – квадратная матрица, состоящая из коэффициентов исходного сигнала.
- $s(t)$  – множество сигналов выделяемых источников.

Имея оценку матрицы  $A$  можно получить искомые источники с помощью обратной ей матрицы  $W$ :

$$y(t) = Wx(t) \quad (1.4)$$

В рамках задачи слепого выделения источников данный метод сводится к серии матричных произведений, представляющих собой фильтры. В общем случае входной сигнал  $X$  размерности  $N$  из  $M$  образцов (представляется матрицей размерности  $N \times M$ ) может быть приведен к сигналу  $Y$  с использова-

нием матрицы преобразования  $W$  размерности  $N \times N$  как  $Y^T = WX^T$ . Такое преобразование проецирует сигнал на разные оси основываясь на матрице преобразования. Если размер полученного сигнала равен размер исходного сигнала, то преобразование называется ортогональным и оси перпендикулярны.

Данный метод является итеративным с фиксированной точкой. В нем применяется нелинейная функция  $g(y) = \text{th}(a * y)$ , которая применяется к матрице разделения  $W$ , которая пересчитывается на каждом шаге.

Входные данные для данного метода должны быть обработаны следующей последовательностью действий:

- а) центрирование по среднему значению;
- б) нормализация по дисперсии;
- в) ортоганализация.

После этого случайнм образом выбирается начальное значение вектора  $W$ , которое уточняется с помощью следующих итеративных шагов:

- а) Обновление  $W$  по формуле:

$$w = E\{zg(w^T z)\} - E\{g'(w^T z)\}w \quad (1.5)$$

где  $z$  – обработанные входные данные.

$$\text{б) } w = \frac{w}{\|w\|}$$

Выполняются до достижения сходимости.

### 1.2.2 Факторизация неотрицательных матриц

Факторизация неотрицательных матриц (англ. Non-Negative Matrix Factorization) широко использовалась в области выделения источников в прошлом. Основная идея данного метода заключается в представлении матрицы  $Y$  в виде комбинации базиса  $B$  и активационного усиления  $G$  как  $Y = BG$ . Базовый вектор представляет частотную характеристику источника в заданный момент времени, а вектор  $G$  представляет усиление частот в любой момент времени. Таким временем  $G$  является горизонтальным вектором вдоль времени.

В контексте задачи выделения аудио источников, если исходный сигнал является объединением двух источников,  $S_1$  и  $S_2$ , так, что  $Y = S_1 + S_2$ , и базисные вектора двух источников вычисляются как  $B_1$  и  $B_2$ , то исходный сигнал можно представить как  $Y = B_1G_1 + B_2G_2$ , где  $G_1$  и  $G_2$  – соответствующие активационные усиления для двух источников, представленные в разные моменты времени.

Для применения факторизации неотрицательных матриц необходимо, чтобы сигнал представлял из себя линейную комбинацию базисных векторов. Для  $K$  источников:

$$X_{i,j} = \sum_{k=1}^K B_{i,k} G_{k,j} \quad (1.6)$$

Расхождение между  $X$  и  $BG$  должно быть минимизировано, чтобы гарантировать, что найденные аудио источники представляют в комбинации исходный сигнал:

$$B, G = \operatorname{argmin}_{B, G \geq 0} D(X, BG) \quad (1.7)$$

где  $D$  является функцией расхождения, которая может быть:

a) среднеквадратичным отклонением:

$$D(A, B) = \|A - B\|^2 = \sum_{ij} (A_{ij} - B_{ij})^2 \quad (1.8)$$

б) дивергенцией Кулбека-Лейблера:

$$D(A, B) = \sum_{ij} (A_{ij} \log \frac{A_{ij}}{B_{ij}} - A_{ij} + B_{ij}) \quad (1.9)$$

Для этой дивергенции применяется алгоритм мультипликативного обновления [13]:

a) Вектора  $B$  и  $G$  заполняются случайными значениями.

б) Вычисление нового значение  $B$ :

Для евклидова расстояния:

$$B \leftarrow B \frac{XG^T}{(BG)G^T} \quad (1.10)$$

Для дивергенции Кулбека-Лейблера:

$$B \leftarrow B \frac{(\frac{X}{BG})G^T}{1G^T} \quad (1.11)$$

в) Вычисление нового значения  $G$ :

Для евклидова расстояния:

$$G \leftarrow G \frac{B^T X}{B^T (BG)} \quad (1.12)$$

Для дивергенции Кулбека-Лейблера:

$$G \leftarrow G \frac{B^T \frac{X}{BG}}{B^T 1} \quad (1.13)$$

Эти действия повторяются предопределенное количество раз, либо до момента достижения дивергенции определенного минимума.

После нахождения магнитуды источника, сигнал можно получить при помощи фазовой характеристики исходного сигнала.

Работа данного метода зависит от задаваемых начальных условий, зависящих от числа источников и фильтров, примененных к итоговому сигналу.

### 1.2.3 Адаптивный метод FASST

Адаптивный метод FASST (англ. Flexible Audio Source Separation Toolbox) являются модификацией метода факторизации неотрицательных матриц, основанный на обобщенном EM-алгоритме (англ. Generalized expectation-maximization (GEM) algorithm) [14].

Суть данного метода заключается в обобщении реализации, гибкой для разных случаев начальных условий. Каждый выделяемый источник представляется в виде модели возбуждающего фильтра:

$$V_j = V_j^{ex} \odot V_j^{ft} \quad (1.14)$$

где  $V_j^{ex}$  представляет спектральный возбудитель мощности и  $V_j^{ft}$  представляет спектральный фильтр мощности,  $\odot$  означает поэлементное умножение матриц.

В дальнейшем  $V_j^{ex}$  определяется в качестве суммы спектральных шаблонов  $E_j^{ex}$ , модулированных временными коэффициентами  $P_j^{ex}$ . Эти параметрами являются аналогами базисного вектора и активационного усиления в методе факторизации неотрицательных матриц. Тем самым  $V_j^{ex}$  можно переписать как:

$$V_j^{ex} = E_j^{ex} P_j^{ex} \quad (1.15)$$

Спектральные шаблоны представляются в виде линейной комбинации произведения узкополосных спектральных шаблонов  $W_j^{ex}$  и неотрицательных весов  $U_j^{ex}$ .

Временные коэффициенты представляются в виде линейной комбинации произведения кратковременных шаблонов  $H_j^{ex}$  и  $G_j^{ex}$ .

В итоге можно записать:

$$V_j = (W_j^{ex} U_j^{ex} G_j^{ex} H_j^{ex}) \odot (W_j^{ft} U_j^{ft} G_j^{ft} H_j^{ft}) \quad (1.16)$$

Эти шаблоны оцениваются для каждого источника с помощью ГЕМ алгоритма, с использованием алгоритма мультипликативного обновления [13].

#### 1.2.4 Методы, использующие нейронные сети

Идея применения методов машинного обучения в задачах разделения аудио источников возникла после успешных использований нейронных сетей в других областях, в частности обработки изображений [15]. Для решения данной задачи наиболее широко рассматривались глубокие[16] и сверточные[6] нейронные сети.

##### 1.2.4.1 Общий принцип работы нейронных сетей

Нейронная сеть – система обработки информации, вдохновленная нервной системой человека[17]. Подобно нервной системе, искусственные нейронные сети состоят из соединения узлов, называемых нейронами. Каждый нейрон получает входной сигнал, обрабатывает информацию во входном сигнале и дает выход. Эти нейроны содержат параметры, которые должны быть

оптимизированы с помощью тренировочного набора, который имеет истинные результирующие значения. После обучения сеть может использоваться для ввода данных с целью получения результатов для тестирования и будущего использования. Схема простой нейронной сети представлена на рисунке 1.2 и состоит из трех вертикальных слоев:

- Входной слой – распределение входных сигналов остальным нейронам. Нейроны этого слоя не производят никаких вычислений.
- Скрытый слой – принимает сигналы с предыдущего слоя, обрабатывает поулучченную информацию и передает результат следующему слою.
- Выходной слой – выполняет схожие со скрытым слоем функции за исключением того, что результатом работы выходного слоя является результат работы всей нейронной сети.

Каждый слой состоит из нейронов, каждый из которых можно описать математически с помощью его параметров:

- Вектор входных данных  $X$ , состоящий из значений  $x_1, \dots, x_n$ .
- Пороговое число  $b$ , является константой, добавляющейся к входному вектору.
- Вектор входных весов  $W$ , состоящий из значений  $w_1, \dots, w_n$ .
- Нелинейная активационная функция  $\sigma$ , которая может быть:

Гиперболический тангенс (рис. 1.3):

$$\sigma(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (1.17)$$

Сигмоида (рис. 1.4):

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (1.18)$$

Линейная функция активации ReLU (рис. 1.22):

$$\sigma(x) = \max(0, x) \quad (1.19)$$

- Выходное значение, которое находится по формуле:

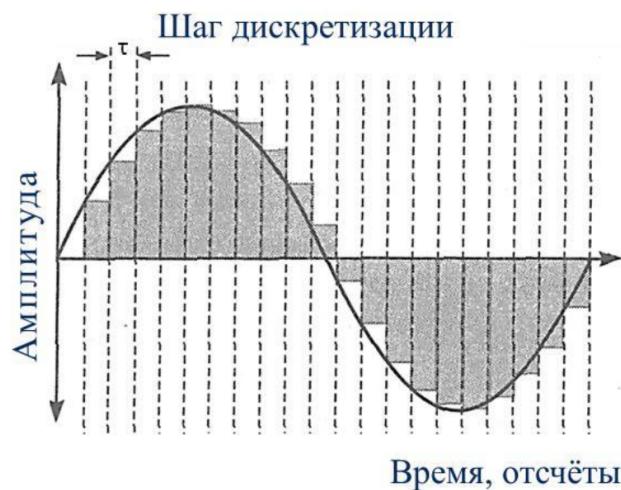


Рисунок 1.1 — Дискретизация аналогового сигнала

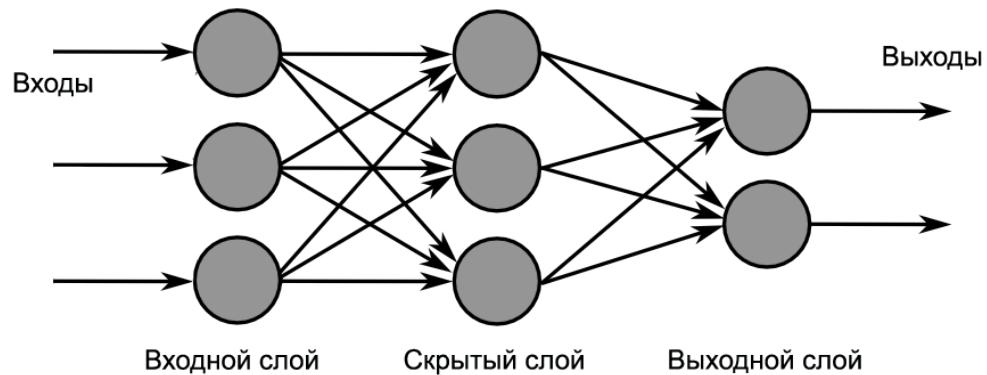


Рисунок 1.2 — Схема простой нейронной сети

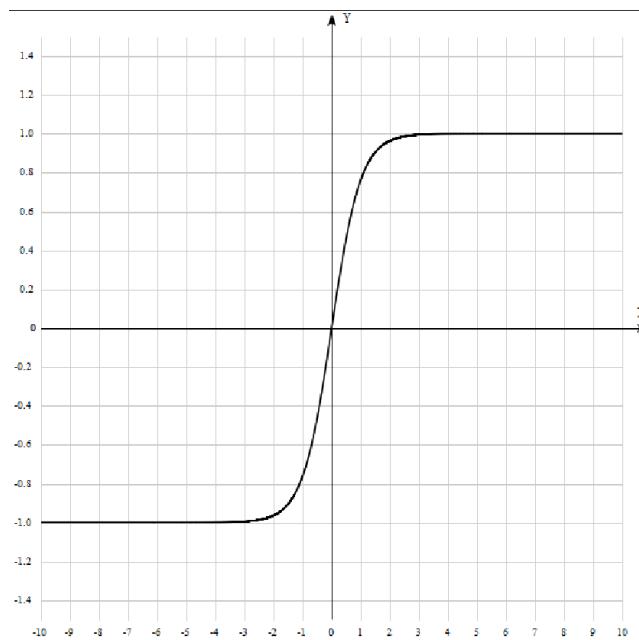


Рисунок 1.3 — График гиперболического тангенса

$$a = \sigma\left(\sum_{i=1}^n w_i x_i + b\right) \quad (1.20)$$

Поскольку каждый узел содержит нелинейную функцию, сеть способна изучать различные нелинейные представления входных данных с помощью различных комбинаций входных узлов и функций активации.

#### 1.2.4.2 Глубинные нейронные сети

Глубинные нейронные сети (англ. Deep neural networks) – нейронные сети с более чем одним скрытым слоем (рис. 1.5).

По мере того как данные проходят через более чем один слой, можно обнаружить более абстрактные представления данных, что может помочь улучшить классификацию данных. Каждый слой глубинной сети имеет входы и выходы, и Количество входов каждого слоя зависит от количества выходов его предшественника. Если вход  $k$  – слоя –  $x_k$ , то его выход можно записать как:

$$a_k = \sigma_k(b_k + \sum_{i=1}^n w_{k,i} x_{k,i}) \quad (1.21)$$

На каждом отдельном слое могут быть использованы отдельные активационные функции.

ГНС рассматривались в качестве инструмента в задачах выделения инструментальных аудио источников[18].

Метод состоит из трех основных шагов:

а) Препроцессинг

Исходный сигнал  $y(n)$  преобразуется в его частотное представление с помощью оконного преобразования Фурье, используя прямоугольную оконную функцию. Из этого представления строится вектор  $x \in \Re^{(2C+1)L}$ , состоящий из суммы значений магнитуды текущего кадра и  $C$  предыдущих/следующих кадров,  $L$  – количество значений магнитуды на кадр.

Полученный вектор нормализуется с помощью числа  $\gamma > 0$ , являющегося средней Евклидовой нормой  $2C + 1$  кадров.

б) Работа глубокой нейронной сети

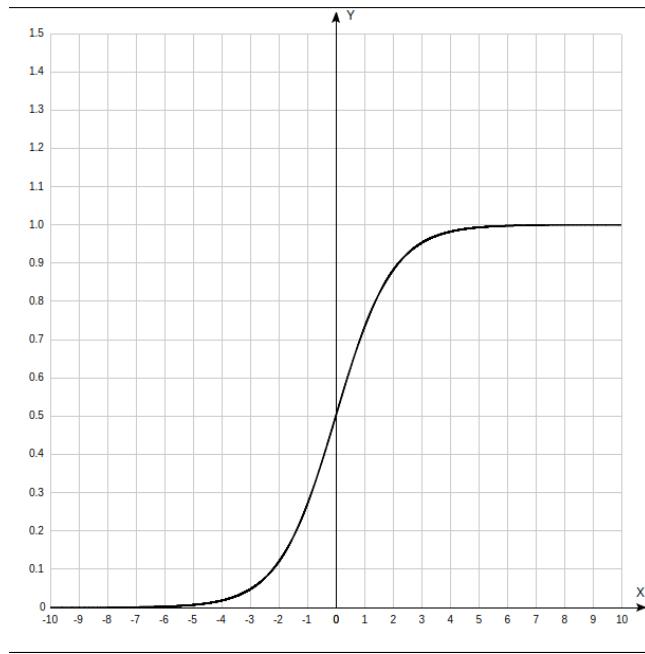


Рисунок 1.4 – График сигмоиды

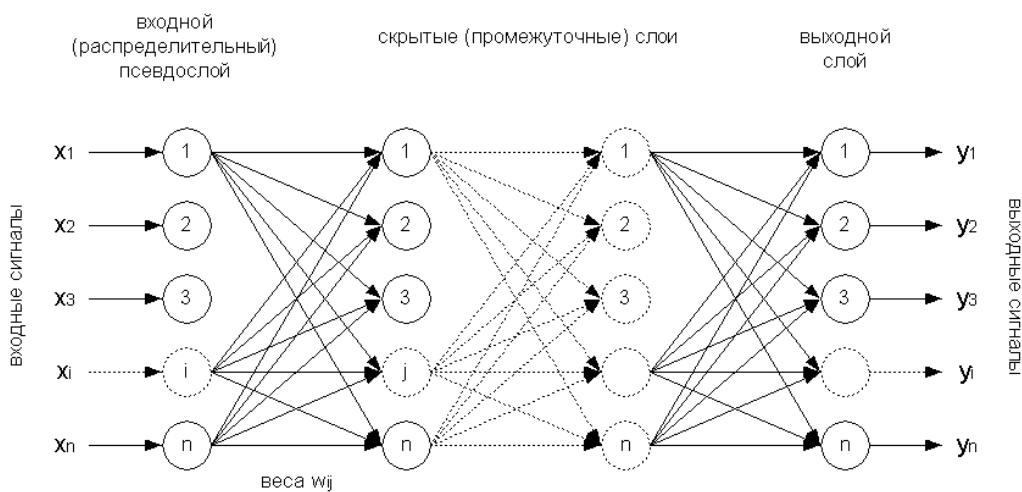


Рисунок 1.5 – Общая схема ГНС

Полученный нормализованный вектор подается на вход DNN, состоящей из  $K$  слоев, с выпрямленной линейной функцией активации (англ. rectified linear unit, ReLU), имеющей вид

$$x_{k+1} = \max(W_k x_k + b_k, 0), k = 1, \dots, K \quad (1.22)$$

где

- $x_k$  — входной вектор для  $k$ -го слоя, в частности  $x_1$  является исходным вектором,  $x_{K+1}$  — результирующий вектор работы ГНС  $\hat{s}$
- $W_k$  — веса  $k$ -го слоя
- $b_k$  — пороговые значения  $k$ -го слоя

Результатом работы ГНС является вектор  $\hat{s}$ , являющийся значением магнитудного окна  $s \in \Re^L$  выделяемого инструмента.

### в) Восстановление

Данный шаг предполагает получение результата применения ОПФ к выделяемому сигналу  $s(n)$  при помощи фазы ОПФ исходного сигнала и произведения каждого полученного вектора  $\hat{s}$  с коэффициентом нормализации  $\gamma$ , используемым выше. Получение итогового сигнала происходит с помощью обратного ОПФ.

Общая схема работы метода представлена на рисунке 1.6.

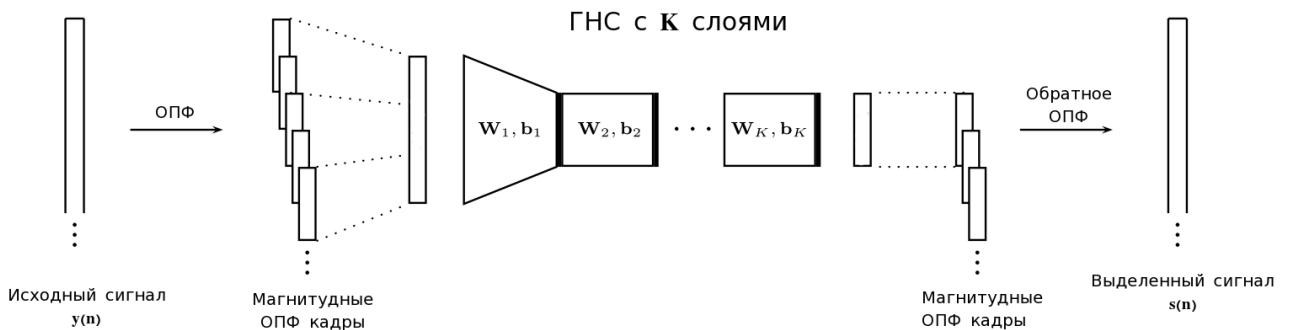


Рисунок 1.6 — Схема работы метода выделения инструментальных аудиосигналов с использованием ГНС

#### 1.2.4.3 Сверточные нейронные сети

Сверточные нейронные сети (англ. Convolutional neural networks) — это вариант нейронных сетей, вдохновленных зрительной корой головного мозга.

га человека[19], используются в основном для обработки изображений. СНС используют локальную пространственную корреляцию между входными нейронами из изображения с помощью локальных сенсорных полей. Для иллюстративных целей входной слой вместо одномерного слоя нейронов можно рассматривать как двумерную матрицу, как в случае с файлом изображения. Для изображений значения этой двумерной матрицы представляют интенсивности пикселей.

В обычной нейронной сети каждый входной нейрон был бы связан с каждый нейроном на первом скрытом слое, в то время как в СНС каждый нейрон первого скрытого слоя связан только с небольшой группой нейронов (рис. 1.7). Сенсорное поле перемещается по всему входному массиву для формирования скрытого слоя. Движение может иметь различный шаг, например на рисунке 1.7 изображен шаг  $(1, 1)$ . Количество нейронов в скрытом слое зависит от количества единиц во входном слое, шага и размера поля.

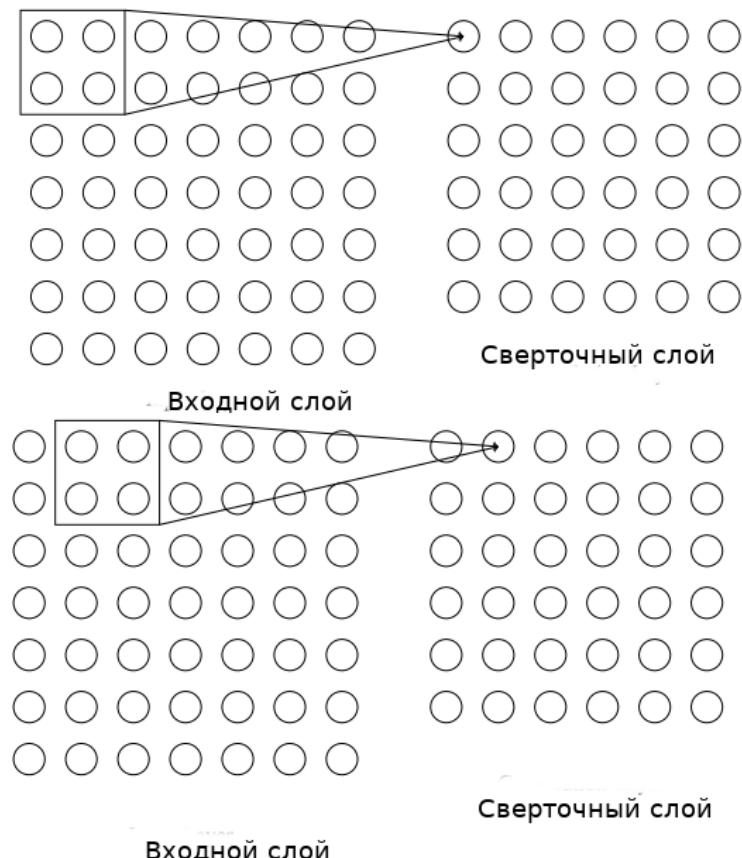


Рисунок 1.7 — Связывание входного слоя с первым скрытым слоем

Если входной слой имеет размерность  $M \times N$ , шаг равен  $(1, 1)$  и сенсорное поле имеет размерность  $A \times B$ , то скрытый слой будет иметь размерность

$(M - A + 1) \times (N - B + 1)$ . Ключевая особенность СНС заключается в том, что нейроны скрытого сверточного слоя разделяют веса и смещения, так что процесс работы сети похож на свертку матрицы размерности  $A \times B$  над матрицей  $M \times N$ .

Результат  $j, k$  – нейрона сверточного слоя математически можно описать как:

$$a_{j,k} = \sigma \left( b + \sum_{l=0}^{A-1} \sum_{m=0}^{B-1} w_{i,m} x_{j+l, k+m} \right) \quad (1.23)$$

Другими словами, сверточный слой вычисляет активацию объекта размерности AxB в разных областях входного слоя. Сопоставление входного слоя с сверточным часто называют векторной картой, а общие веса и единицы смещения называются ядром. Так как каждое из этих ядер идентифицирует одну особенность во входном слое, то сверточный слой обычно включает в себя более одного ядра или карты признаков.

В контексте задачи разделения источников СНС используют для реализации основной идеи ФНМ. В то время, как ФНМ алгоритму необходимо находить базисные вектора для каждой ноты аудиоисточника, рассматриваемый алгоритм использует СНС с целью унификации этих действий через вычисление меньших устойчивых тембральных структур на меньших частях спектограммы. Общая схема работы алгоритма представлена на рисунке 1.8.

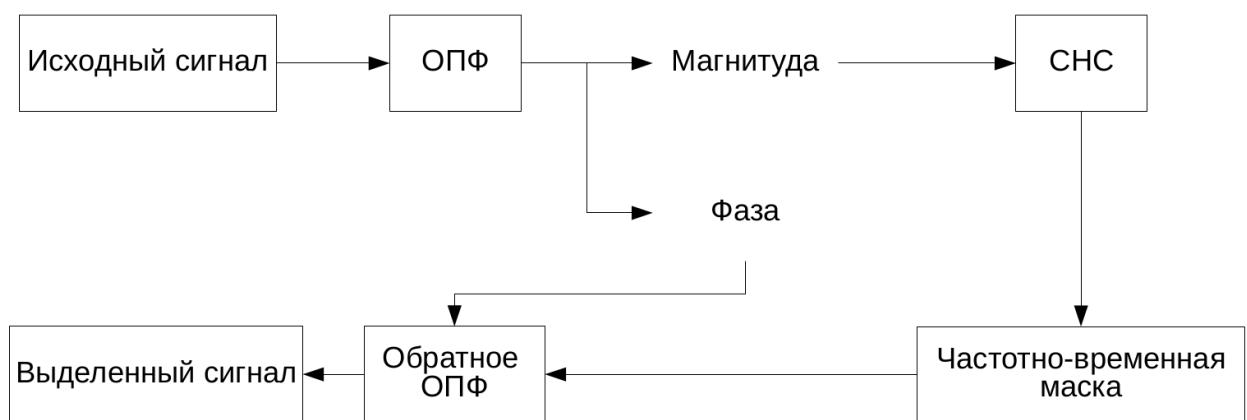


Рисунок 1.8 – Схема работы метода выделения аудио сигналов источников с использованием СНС

На вход СНС подается магнитудная спектограмма, являющаяся результатом применения ОПФ к исходному сигналу.

Работа нейронной сети происходит в четыре этапа:

а) Свертка

Состоит из двух сверточных слоев и объединяющего слоя.

- 1) Тембральный сверточный слой: получает локальную информацию о тембре, позволяя модели изучать особенности тембра. В отличие от подхода ФНМ, когда для каждого источника определяются конкретные базисные и активационные вектора, эти характеристики являются общими для отдельных источников.
- 2) Объединяющий слой: сжимает информацию, указанную в первом слое по частоте и времени для достижения компактного представления данных. На данном слое сравниваются результаты трех типов: без уплотнения, с уплотнением по частоте и с уплотнением по частоте и времени одновременно.
- 3) Временный сверточный слой: изучает различные типы временных изменений для разных инструментов на основании характеристик, полученных из тембрального сверточного слоя. Результатом являются частотно-временные характеристики различных аудио источников.

б) Сокращение размерности

Данный слой состоит из нелинейной комбинации признаков, полученных из предыдущих слоев, с ReLU. Слой выбирается таким образом, чтобы иметь меньше элементов для уменьшения общих параметров сети и обеспечения того, чтобы сеть не перегружала данные и могла создавать надежное представление данных.

в) Обратная свертка

Проводятся действия, являющиеся обратными операции свертки. Состоит из последовательной развертки и масштабирования данных.

г) Частотно временное экранирование

На данном этапе на основе полученных данных вычисляется частотно-временная маска для каждого выделяемого источника:

$$m_n t(f) = \frac{|\hat{y}_{nt}(f)|}{\sum_{n=1}^N |\hat{y}_{nt}(f)|} \quad (1.24)$$

где  $\hat{y}_{nt}(f)$  – результат работы СНС для  $n$ -го источника в момент времени  $t$ ,  $N$  – число выделяемых источников.

Полученная маска применяется к исходному сигналу, в результате получая спектрограмму выделяемого сигнала:

$$y_n(f) = m_n t(f) x_t(f) \quad (1.25)$$

где  $x_t(f)$  – спектрограмма исходного сигнала.

Как и в случае с ГНС, для вычисления выделяемого сигнала, к полученной спектрограмме применяется обратное ОПФ.

#### 1.2.4.4 Рекуррентные нейронные сети

Поскольку аудио сигналы имеют временной контекст, нейронной сети должна быть предоставлена некоторая память для добавления контекстной информации из прошлого. Одним из решений является использование рекуррентной нейронной сети. Суть заключается в соединении скрытого слоя с самим собой (рисунок 1.9). То есть вход слоя в момент времени  $t$  можно описать так:

$$J(j,t) = \sum X(i,t) W_1(i,j) + b_1 W_1(i+1,j) + W_2 H(J(j,t-1)) \quad (1.26)$$

где  $H(J(j,t-1))$  – выход слоя в момент времени  $t-1$ .

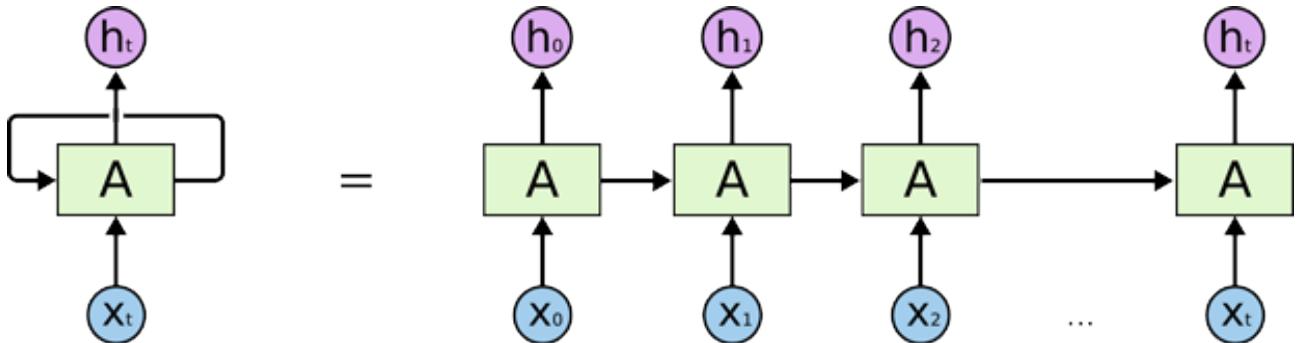


Рисунок 1.9 – Схема однолойной рекуррентной нейронной сети

Существуют так же многослойные рекуррентные сети. Различают многослойные РНС с одной рекуррентной связью на  $l$ -слое и сложенные многослойные РНС, с рекуррентными связями на каждом скрытом слое (рисунок 1.10).

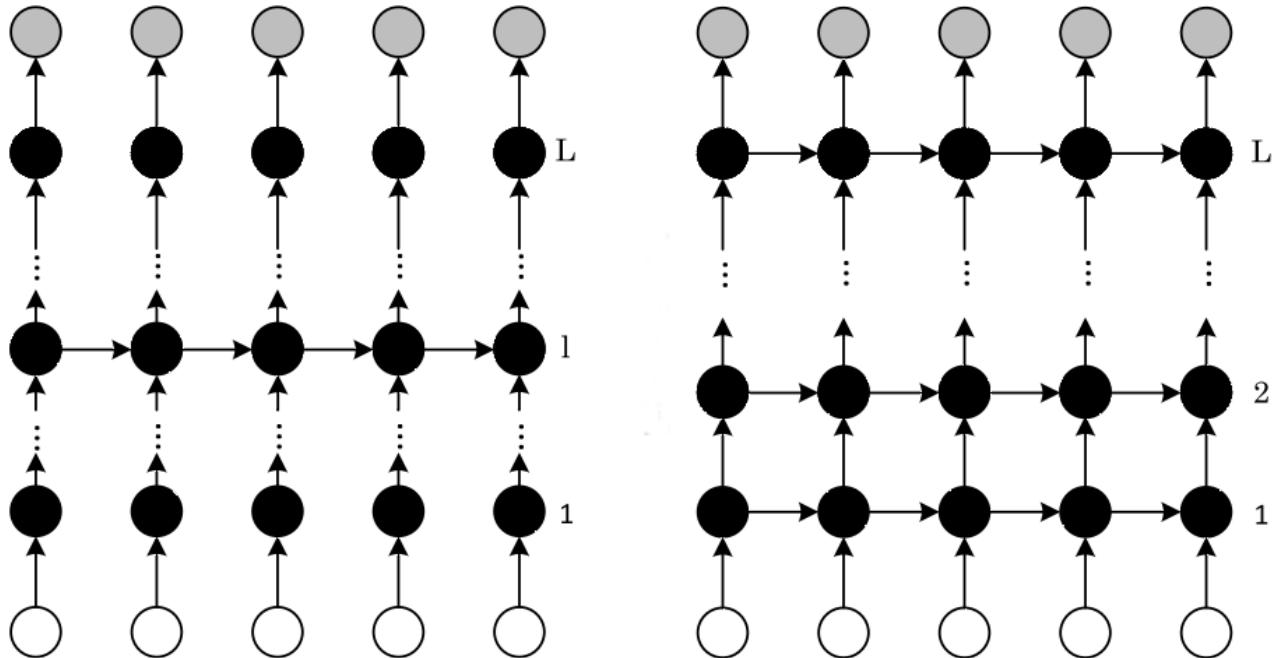


Рисунок 1.10 — Схемы многослойной РНС с одной рекуррентной связью и сложенной многослойной РНС, с  $L$  скрытыми слоями

Эта работа будет сфокусирована на использовании сложенной многослойной РНС. На вход сети будет подаваться спектrogramма исходного сигнала, полученная из ОПФ. Сеть должна генерировать частотно-временную маску, которая на последнем слое будет применяться к спектrogramме исходного слоя.

### 1.3 Оценка методов выделения источников

Оценка методов выделения источников является нетривиальной задачей, учитывая субъективность мнения о качестве аудио записи. Тем не менее, оценочные метрики были изучены и формализованы группой исследователей во главе с Имануилом Винсентом[20]. Оценка метода выделения аудио источника должна проходить в контексте приложения. Исходный сигнал представляется линейной комбинацией сигналов источников  $(s_j)_{j \in J}$  с добавлением шума

ма. Простейшей мерой является норма L2, выражающаяся в разнице между полученным источником  $\hat{s}_j$  и ожидаемым источником  $s_j$ :

$$D = \min_{\epsilon=\pm 1} \left\| \frac{\hat{s}_j}{\|\hat{s}_j\|} - \epsilon \frac{s_j}{\|s_j\|} \right\|^2 \quad (1.27)$$

Данная разность всегда неотрицательна и равна нулю только в случае  $\hat{s}_j = s_j$ . В худшем случае, когда вектора  $\hat{s}_j$  и  $\hat{s}$  ортогональны, разность останется ограниченной максимумом двух, так как источники нормализованы. Так же, в случае шумов или других искажений, которые добавляются к искомым источниками, данная мера будет иметь низкое значение, которое может быть нежелательным, в зависимости от приложения. Например, при обработке аудиозаписей высокого качества, такие шумы могут быть нежелательными, а в случае, когда к выделяемым источникам планируется применять дополнительную, данными искажениями можно пренебречь.

Для учета этих случаев искомый источник разделяется на четыре части:

$$\hat{s}_j = s_{target} + e_{interf} + e_{spat} + e_{artif} \quad (1.28)$$

где

- $s_{target}$  — модификация целевого источника, которая может содержать определенные допустимые искажения,  $F$ ;
- $e_{interf}$  — помехи, полученные от нежелательных источников  $(s_{j'})_{j' \neq j}$ ;
- $e_{spat}$  — пространственные искажения;
- $e_{artif}$  — остальные виды шумов, не входящие в  $F$  и другие артефакты.

Для определения метрик используются ортогональные проекции, которые могут быть определены следующим образом: для подпространства с векторами  $y_1, y_2, \dots, y_n$ ,  $\prod y_1, y_2, \dots, y_n$  — матрица, проектирующая вектора в данное подпространство. Определяются следующие матрицы:

$$P_{s,j} = \prod (s_j) \quad (1.29)$$

$$P_S = \prod \{(s_{j'})_{1 \leq j' \leq n}\} \quad (1.30)$$

$$P_{S,n} = \prod \{(s_{j'})_{1 \leq j' \leq n}, (n_i)_{1 \leq i \leq m}\} \quad (1.31)$$

С помощью этих проецирующих матриц слагаемые из формулы 1.28 можно представить следующим образом:

$$s_{target} = P_{s,j} \hat{s}_j \quad (1.32)$$

$$e_{interf} = P_S \hat{s}_j - P_{s,j} \hat{s}_j \quad (1.33)$$

$$e_{spat} = P_{S,n} \hat{s}_j - P_{s,j} \hat{s}_j \quad (1.34)$$

$$e_{artif} = \hat{s}_j - P_{S,n} \hat{s}_j \quad (1.35)$$

Для вычисления  $s_{target}$  используется скалярное произведение:

$$s_{target} = (\hat{s}_j, s_j) \frac{s_j}{\|s_j\|^2} \quad (1.36)$$

Для вычисления  $P_S$  и  $P_{S,n}$  используется вектор коэффициентов  $c$ :

$$c = R_{SS}^{-1} [(\hat{s}_j, s_1) \dots (\hat{s}_j, s_n)]^H \quad (1.37)$$

где  $R_{SS}$  – определитель Грама (грамиан).  $P_S \hat{s}_j$  можно записать как:

$$P_S \hat{s}_j = c^H s \quad (1.38)$$

где  $(.)^H$  – Эрмитовская перестановка.

Если источник и шумовой сигнал взаимно ортогональны, то можно записать:

$$e_{interf} = \sum_{j' \neq j} (\hat{s}_j, s_{j'}) \frac{s_{j'}}{\|s_{j'}\|^2} \quad (1.39)$$

$$P_{S,n} \hat{s}_j = P_S \hat{s}_j + \sum_{i=1}^m (\hat{s}_j, n_i) \frac{n_i}{\|n_i\|^2} \quad (1.40)$$

На основании этих значений можно записать следующие метрики:

а) Отношение источника к искажению:

$$SDR = 10 \log_{10} \frac{\|s_{target}\|^2}{\|e_{interf} + e_{spat} + e_{artif}\|^2} \quad (1.41)$$

Эта мера оценивает общую эффективность алгоритма разделения источников

б) Отношение источника к интерференции:

$$SIR = 10 \log_{10} \frac{\|s_{target}\|^2}{\|e_{interf}\|^2} \quad (1.42)$$

Эта мера отражает подавление помех во время выделения.

в) Отношение образа к пространственному искажению:

$$ISR = 10 \log_{10} \frac{\|s_{target} + e_{interf}\|^2}{\|e_{spat}\|^2} \quad (1.43)$$

г) Отношение источника к артефактам:

$$SAR = 10 \log_{10} \frac{\|s_{target} + e_{interf} + e_{spat}\|^2}{\|e_{artif}\|^2} \quad (1.44)$$

Эта мера оценивает артефакты, получаемые во время разделения источников.

Значение компонент  $s_{target}$ ,  $e_{interf}$ ,  $e_{spat}$  и  $e_{artif}$  изменяются во времени. Предлагается разделять сигнал на части для вычисления локальных значений. Итоговые значения могут быть суммированы с использованием статистических мер.

## 1.4 Сравнение методов

Метод главных компонент и Факторизация неотрицательных матриц не участвуют в сравнении, так как являются строго статистическими методами и требуют дополнительных действий для подбора начальных значений для каждого конкретного примера.

Значение метрик получено на наборе данных DSD100 [21]. Как видно из таблицы, использование нейронных сетей дает лучший результат в задаче выделения аудио источников.

Таблица 1.1 — Сравнение алгоритмов выделения источников

Метод	Метрика	Вокал	Аккомпанемент
CNN	SDR	-0.6 ± 4.9	4.5 ± 1.2
	SIR	1.9 ± 6.2	14.9 ± 5.6
	SAR	3.6 ± 2.1	16.4 ± 2.9
	ISR	7.3 ± 2.7	6.9 ± 1.0
DNN	SDR	-8.9 ± 4.4	4.1 ± 2.6
	SIR	10.8 ± 3.1	10.7 ± 3.8
	SAR	-6.6 ± 4.6	12.1 ± 3.9
	ISR	4.8 ± 2.8	5.0 ± 3.0
FASST	SDR	-1.8 ± 5.8	8.9 ± 3.2
	SIR	-3.7 ± 7.5	13.7 ± 3.0
	SAR	4.3 ± 1.4	13.1 ± 3.2
	ISR	5.6 ± 2.5	13.5 ± 2.5

## 1.5 Вывод

Необходимо разработать и реализовать метод выделения голосовой составляющей из монофонического аудио сигнала на основе многослойной РНС.

Для выполнения работы необходимо спроектировать и реализовать продукт, осуществляющий автоматизированное выделение вокальной составляющей из музыкального аудио сигнала. На вход программному продукту подаются одноканальные музыкальные произведения в формате WAVE. На выходе программный продукт генерирует два одноканальных WAVE-файла, содержащие вокальный аудио сигнал и аккомпанемент. Исходные музыкальные произведения должны представлять из себя песни с эстрадным стилем вокала. Разделяемые записи должны представлять собой записи «без потерь», на них не должно быть шумов. Для более точного разделения запись должна быть без фильтров, применяемых на этапе пост-обработки.

Для определения качества выделения источников будут использоваться сравнение полученных в результате работы алгоритмов данных с заранее известными на основе описанных выше метрик.

Так же необходимо сравнить показатели разработанного метода с значениями, описанными в таблице 1.1.

## 2 Конструкторский раздел

### 2.1 Архитектура программного продукта

Разрабатываемый программный продукт состоит из следующих частей:

- а) модуль обработки входных данных;
- б) модуль генерации магнитуд для вокала и аккомпанемента;
- в) модуль формирования выходных данных.

IDEF0 диаграмма программы показаны на рисунке 2.1.

### 2.2 Алгоритмы

Алгоритм 1 описывает процесс выделения вокальной составляющей из аудио сигнала. Подробное описание представлено ниже.

**Входные данные:** Аудио сигнал  $x(t)$

**Выходные данные:** Аудио сигнал голосовой составляющей

$$y_1(t), \text{ аудио сигнал аккомпанимента } y_2(t)$$

**Исходные параметры:** Оконная функция  $w$

Получение спектrogramмы с помощью ОПФ:

$$s(f, n) \leftarrow \text{ОПФ}(x(t), w)$$

Получение амплитудной и фазовой характеристик спектограммы:

$$m(f, n) \leftarrow |s(f, n)|$$

$$p(f, n) \leftarrow \text{angle}(s(f, n))$$

Получение амплитудных характеристик спектrogramм

выделяемых источников:

$$\tilde{y}_1(f, n), \tilde{y}_2(f, n) \leftarrow DRNN(m(f, n))$$

Получение сигналов выделяемых источников с помощью

обратного ОПФ:

$$y_1(t) \leftarrow \text{ООПФ}(\tilde{y}_1(f, n), p(f, n))$$

$$y_2(t) \leftarrow \text{ООПФ}(\tilde{y}_2(f, n), p(f, n))$$

**Алгоритм 1:** Алгоритм выделения аудио источников

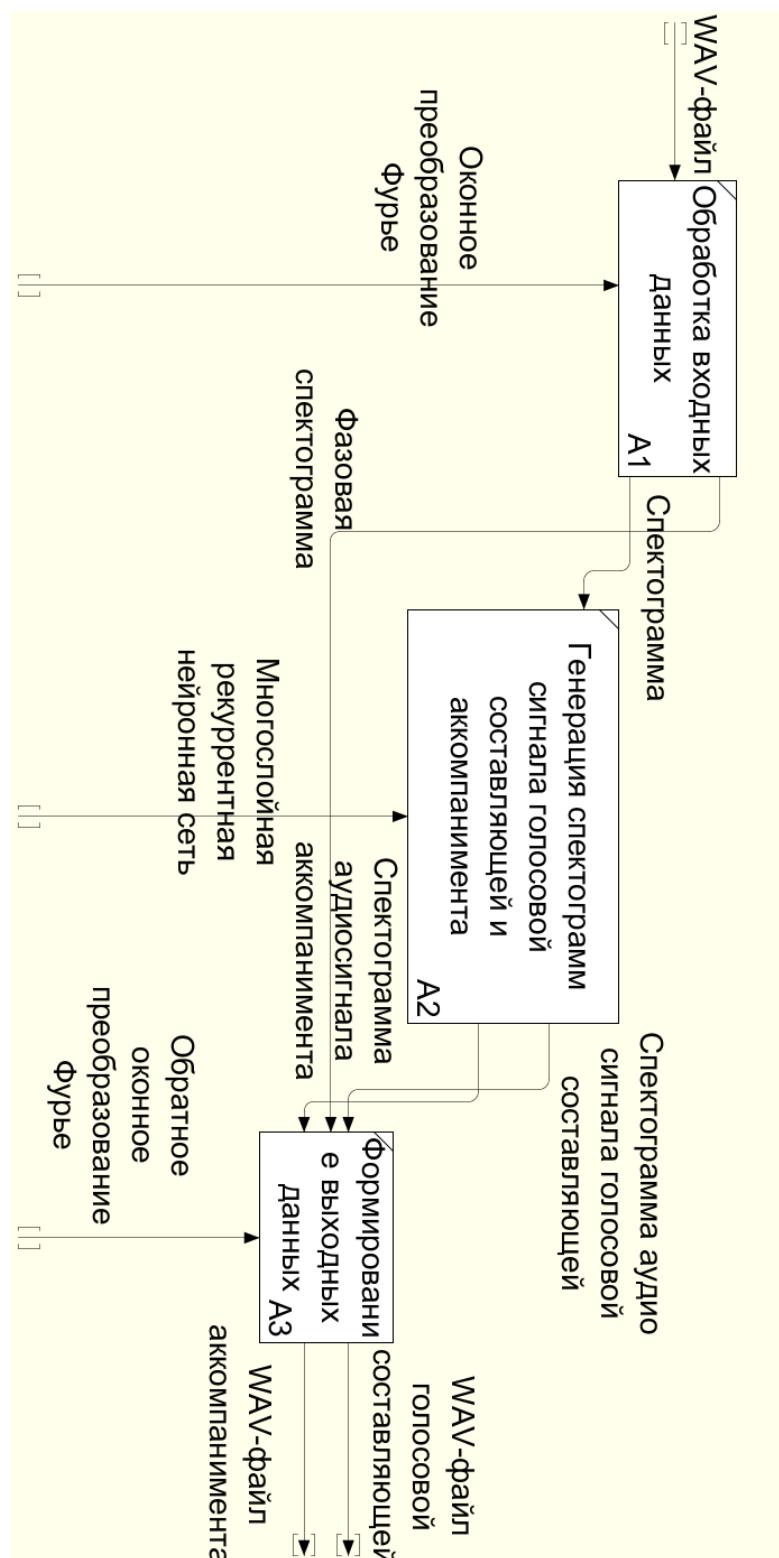


Рисунок 2.1 — IDEF0 программного продукта

### 2.2.1 Обработка входных данных

Чтение аудио-файлов, хранящиеся в формате WAVE, происходит с помощью библиотеки. Результатом работы библиотечной функции является информация о частоте дискретизации и структура со значениями амплитуд сигнала.

После считывания аудио-файла происходит построение его спектра при помощи оконного преобразования Фурье. Результатом работы этого алгоритма является структура, в которой хранятся наборы частот по отрезкам времени.

В качестве оконной функции используется Окно Ханна, дискретная функция которого записывается следующим образом:

$$w(n) = 0.5 - 0.5 \cos\left(\frac{2\pi n}{N-1}\right) \quad (2.1)$$

### 2.2.2 Генерация частотно-временных масок

Для получения частотно-временных масок используется многослойная рекуррентная нейронная сеть. На ее вход подается результат применения ОПФ к исходному сигналу. Результатом работы сети являются спектрограммы выделяемых источников.

### 2.2.3 Описание сети

Многослойная РНС состоит из 6 слоев:

- a) Входной слой. На вход подается спектрограмма исходного сигнала, полученная в следствии применения ОПФ.
- б) 3 скрытых слоя  $h_t^1, h_t^2$  и  $h_t^3$ .

Слой  $h_t^2$  имеет рекуррентную связь. Отсюда результатом работы этого слоя в момент времени  $t$  можно записать как:

$$h_t^2 = f_h(x_t, h_{t-1}^2) = \sigma_2(U^2 h_{t-1}^2 + W^2(h_t^1)) \quad (2.2)$$

где:

- $W^i$  — веса связей, идущих от предшествующего слоя;

- $U^i$  – веса рекуррентной связи;
- $h_t^{i-1}$  – результат работы предшествующего слоя;
- $\sigma_i$  – активационная функция.

Так как у первого слоя предшествующий слой – входной, то его результат вычисляется по следующей формуле:

$$h_t^1 = \sigma_1(W^1 x_t) \quad (2.3)$$

Результатом работы слоя  $h_t^3$  будет:

$$h_t^3 = \sigma_3(W^2 h_t^2) \quad (2.4)$$

В качестве активационной функции используется ReLU (формула 1.22).

в) Слой генерации частотно-временных масок. Данный слой использует результат работы слоя третьего рекуррентного слоя для генерации двух векторов  $\hat{y}_{1t}$  и  $\hat{y}_{2t}$ , размер которых равен размеру входного вектора. Данные последовательности используются для вычисления частотно-временной маски  $m_t(f)$ , которая является функцией частоты:

$$m_t(f) = \frac{|\hat{y}_{1t}(f)|}{|\hat{y}_{1t}(f)| + |\hat{y}_{2t}(f)|} \quad (2.5)$$

г) Слой генерации спектrogramм. Для получения спектrogramм выделяемых источников к спектrogramме исходного аудио-сигнала применяется частотно-временная маска, полученная на предыдущем слое. В итоге получаются два вектора  $\tilde{y}_{1t}$  и  $\tilde{y}_{2t}$ , вычисляющиеся как:

$$\tilde{y}_{1t}(f) = m(f)x_t(f) \quad (2.6)$$

$$\tilde{y}_{2t}(f) = (1 - m(f))x_t(f) \quad (2.7)$$

С учетом формулы 2.5:

$$\tilde{y}_{1t} = \frac{|\hat{y}_{1t}|}{|\hat{y}_{1t}| + |\hat{y}_{2t}|} \odot x_t \quad (2.8)$$

$$\tilde{y}_{2t} = \frac{|\hat{y}_{2t}|}{|\hat{y}_{1t}| + |\hat{y}_{2t}|} \odot x_t \quad (2.9)$$

Схема нейронной сети представлена на рисунке 2.2.

### 2.2.3.1 Обучение нейронной сети

Для обучения нейронной сети используется Метод обратного распространения ошибки. Для вычисления ошибок между сгенерированными векторами  $\tilde{y}_{1t}, \tilde{y}_{2t}$  и ожидаемыми  $y_{1t}, y_{2t}$  используются среднеквадратичное отклонение

$$J_{MSE} = \|\tilde{y}_{1t} - y_{1t}\|^2 + \|\tilde{y}_{2t} - y_{2t}\|^2 \quad (2.10)$$

и дивергенция Кулбека-Лейблера

$$J_{KL} = D(y_{1t}, \tilde{y}_{1t}) + D(y_{2t}, \tilde{y}_{2t}) \quad (2.11)$$

где  $D$  вычисляется по формуле 1.9.

### 2.2.4 Формирование выходных данных

Для формирования WAV-файлов используется обратное оконное преобразование Фурье. В качестве параметров передается матрица, полученная из спектрограммы выделяемого источника и фазовой спектрограммы исходного сигнала.

## 2.3 Используемые структуры данных

Для хранения данных WAV-файла используется последовательность действительных чисел, хранящая информацию об уровне сигнала в определенный момент времени.

Для хранения результатов ОПФ и работы нейронной сети используется матрица комплексных чисел, так как требуется быстрый доступ к элементам, а изменений размеров после первичного заполнения не происходит.

Комплексное число представляет собой структуру с двумя полями: реальной и мнимой частями.

В качестве входных данных нейронной сети используется амплитудная характеристика спектрограммы, хранящаяся в виде матрицы действительных чисел фиксированного размера.

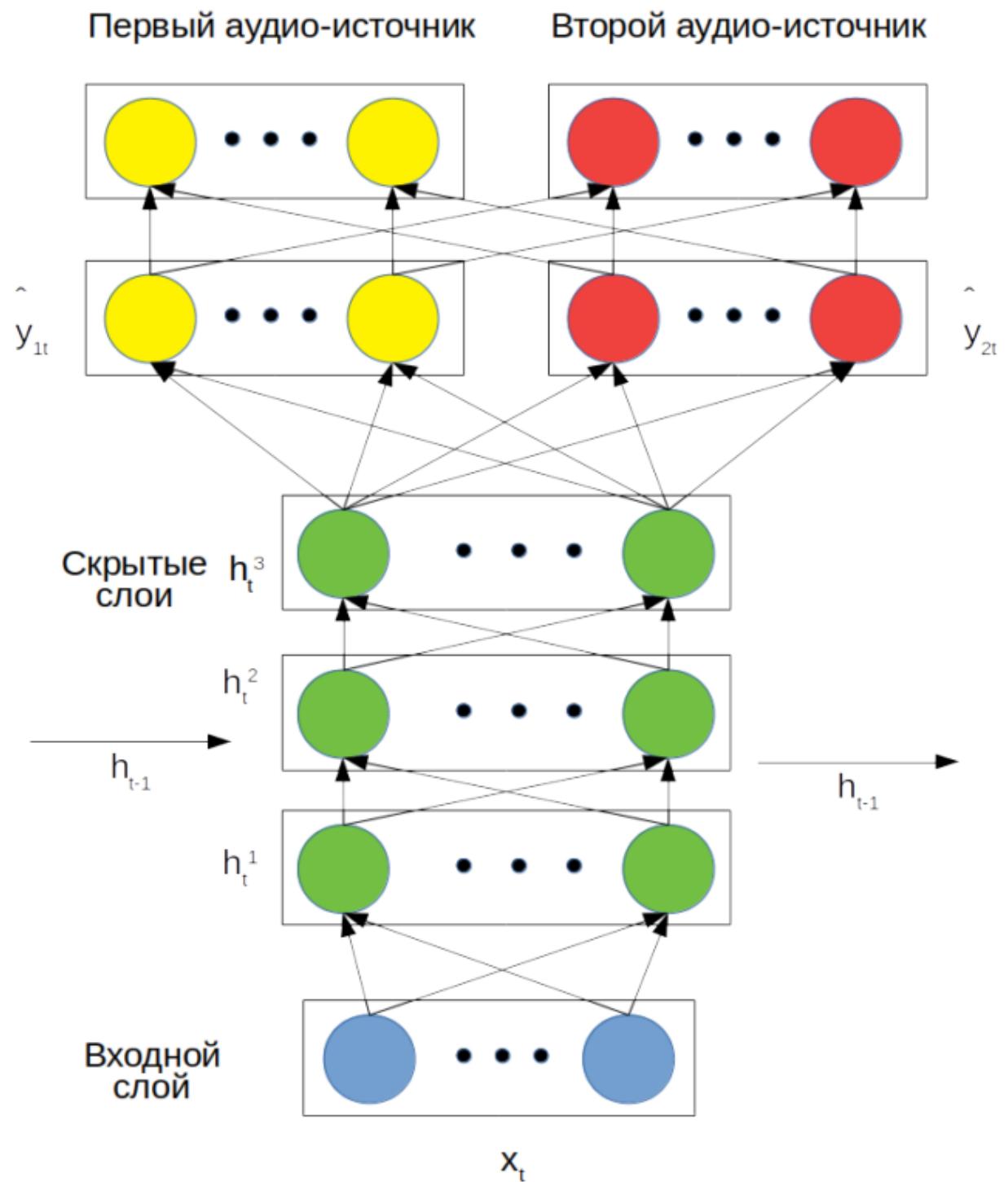


Рисунок 2.2 — Схема нейронной сети

## **2.4 Тестирование**

Тестирование программного продукта заключается в сравнении результата работы методов на тестовых наборах данных с заранее заданным результатом. Тестовые наборы данных (англ. dataset) для проверки работы алгоритмов, связанных с обработкой музыкальных произведений, предоставляются в открытом доступе в Интернете.

Важной характеристикой входных данных является стиль вокала, так как разные стили имеют разные спектральные характеристики. Можно выделить следующие классы эквивалентности:

- академический вокал(классический, оперный);
- эстрадный вокал;
- джазовый вокал;
- рок вокал;
- народное пение;
- горловое пение.

Следует протестировать работу алгоритмов на музыкальных произведениях с эстрадным вокалом.

### **2.4.1 Тестовые данные**

Для тестирования программного продукта используется открытый набор данных DSD100 (Community-Based Signal Separation Evaluation Campaign 2016). В DSD100 содержатся 100 музыкальных произведений разных жанров, с мужским и женским вокалом. Каждый музыкальный трек был сведен с использованием профессиональных средств звукозаписи.

DSD100 состоит из WAV-файлов сведенных музыкальных произведений, хранящихся в папке «Mixture», и WAV-файлов источников, хранящихся в папке «Sources».

Источники разделены на 4 типа:

- вокал;
- бас;
- ударные;

— остальное.

Все записи являются полифоническими (двух-канальными).

Для тестирования разработанного метода данный датасет необходимо привести к следующему виду. Все источники приводятся в монофонический формат. Бас, ударные и остальное объединяются в один WAV-файл.

## 2.5 Вывод

В конструкторском разделе описывается предлагаемый метод выделения голосовой составляющей из монофонического аудио сигнала. Дано подробное описание, построены схемы выбранных алгоритмов. При этом выделены основные этапы работы метода с указанием необходимых исходных данных для его работы и полученных результатов на каждом этапе

### **3 Технологический раздел**

В данном разделе описываются средства, используемые для разработки программного продукта, требования для функционирования ПО, описываются результаты тестирования программного продукта.

#### **3.1 Выбор средств разработки**

##### **3.1.1 Выбор языка программирования**

Для написания программного продукта используется язык Python, так как для него существует множество готовых решений для работы с нейронными сетями. Так же выбранный язык сочетает в себе возможности функционального, структурного и объектно-ориентированного подходов, что позволяет кратко описывать математические конструкции, необходимые для решения поставленной задачи. Так же для данного языка существует большое количество различных математических библиотек, упрощающие работу с комплексными числами, и библиотек для обработки цифровых сигналов.

##### **3.1.2 Выбор среды программирования и отладки**

В качестве среды разработки для языка Python была выбрана кроссплатформенная IDE PyCharm. Предоставляет средства для анализа кода, графический отладчик, инструмент для запуска юнит-тестов. На данный момент PyCharm является бесплатным для образовательных учреждений и проектов с открытым исходным кодом.

Выбор данной среды разработки обусловлен следующими предоставляемыми возможностями, упрощающими разработку приложения и способствующими повышению качества исходного кода:

- статический анализ кода;
- встроенный отладчик;
- навигация по проекту и исходному коду;
- рефакторинг;
- поддержка систем контроля версий.

### **3.1.3 Используемые библиотеки**

Для упрощения реализации математических операций используется библиотека NumPy. Для считывания и записи аудио-файлов, а так же обработки сигналов используется библиотека LibROSA. Для работы с нейронной сетью используется библиотека TensorFlow. Для создания интерфейса используется фреймворк PyQt, который является Python-версией C++ фреймворка Qt.

## **3.2 Система контроля версий**

В процессе разработки программы использовалась система контроля версий Git. Система контроля версий позволяет вносить в проект атомарные изменения, направленные на решения каких-либо задач. В случае обнаружения ошибок или изменения требований, внесенные изменения можно отменить. Кроме того, с помощью системы контроля версий решается вопрос резервного копирования.

Особенности Git:

- данная система контроля версий является децентрализованной, что позволяет иметь несколько независимых резервных копий проекта;
- поддерживается хостингом репозиториев GitLab;
- поддерживается средой разработки PyCharm;
- предоставляет широкие возможности для управления изменениями проекта и просмотра истории изменений.

## **3.3 Требования к вычислительной системе**

Для запуска программы необходимо иметь установленный на ЭВМ интерпретатор для Python 3.6 с установленными библиотеками.

Так как выбранный язык программирования является кроссплатформенным, то требований к использованию операционной системы нет.

Алгоритмы работают с большими объёмами комплексных данных, поэтому объём оперативной памяти компьютера не должен быть меньше 1 ГБ, желательна архитектура x64 (x86-64).

### **3.4 Формат данных**

Входом и выходом программного продукта являются WAV-файлы. Формат WAVE имеет четкую структуру, описанную в [8].

WAV – формат файла-контейнера для хранения записи оцифрованного аудиопотока, в котором для кодирования амплитуды выделяется фиксированное число бит.

WAV-файл состоит из двух частей. Одна из них – заголовок файла, другая – область данных. В заголовке файла хранится информация о:

- размере файла;
- количестве каналов;
- частоте дискретизации;
- количестве бит в сэмпле (глубине звучания).

Длина заголовка составляет 44 байта. Область данных представляет собой набор амплитуд.

В программе используется информация о количестве каналов и частоте дискретизации, а также амплитуды.

### **3.5 Диаграмма классов**

Разработанный программный комплекс имеет два режима работы: режим обучения и нормальный режим. Диаграммы классов представлены на рисунках 3.1 и 3.2 соответственно.

#### **3.5.1 Описание классов**

##### **DataLoader**

Отвечает за загрузку wav-файлов. Описание методов представлено в таблице 3.1.

##### **Diff**

Представляет ошибку, получаемую в процессе обучения и ее изменение. Описание методов представлено в таблице 3.2.

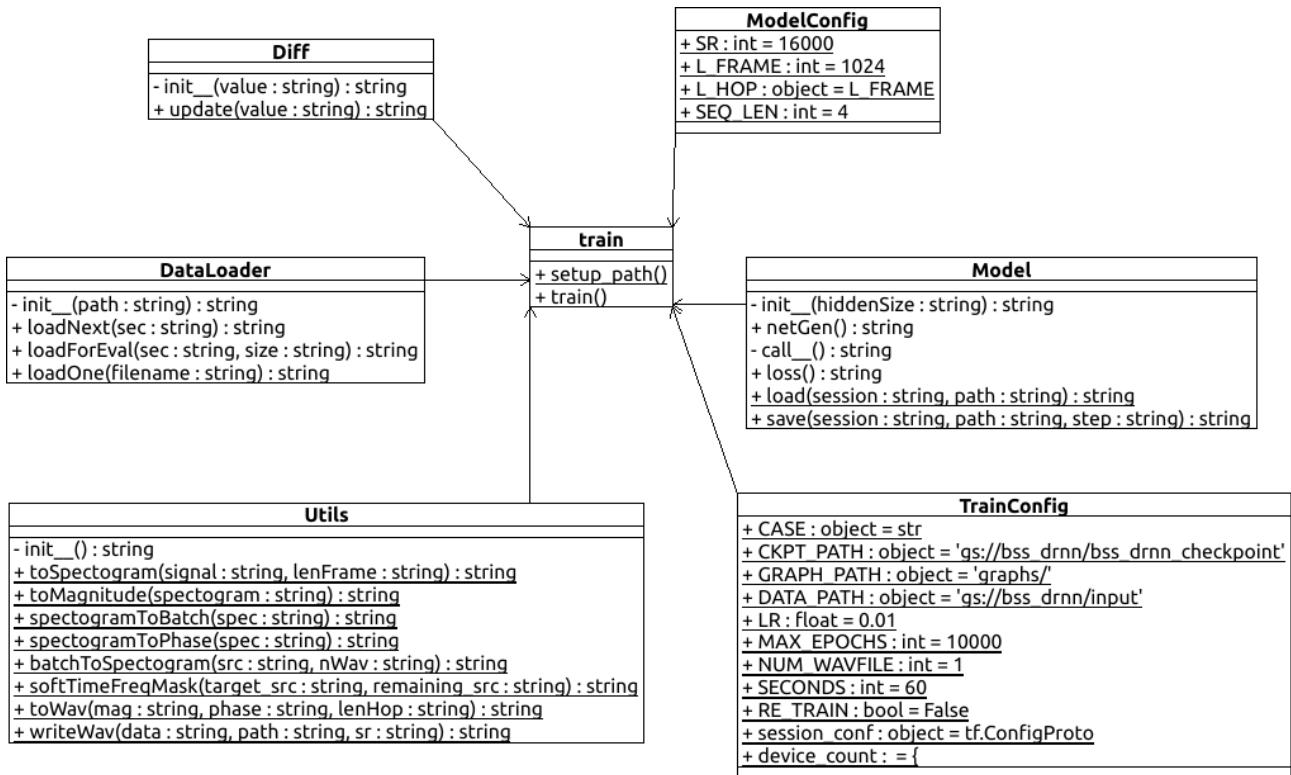


Рисунок 3.1 — Диаграмма классов программы нормального режима

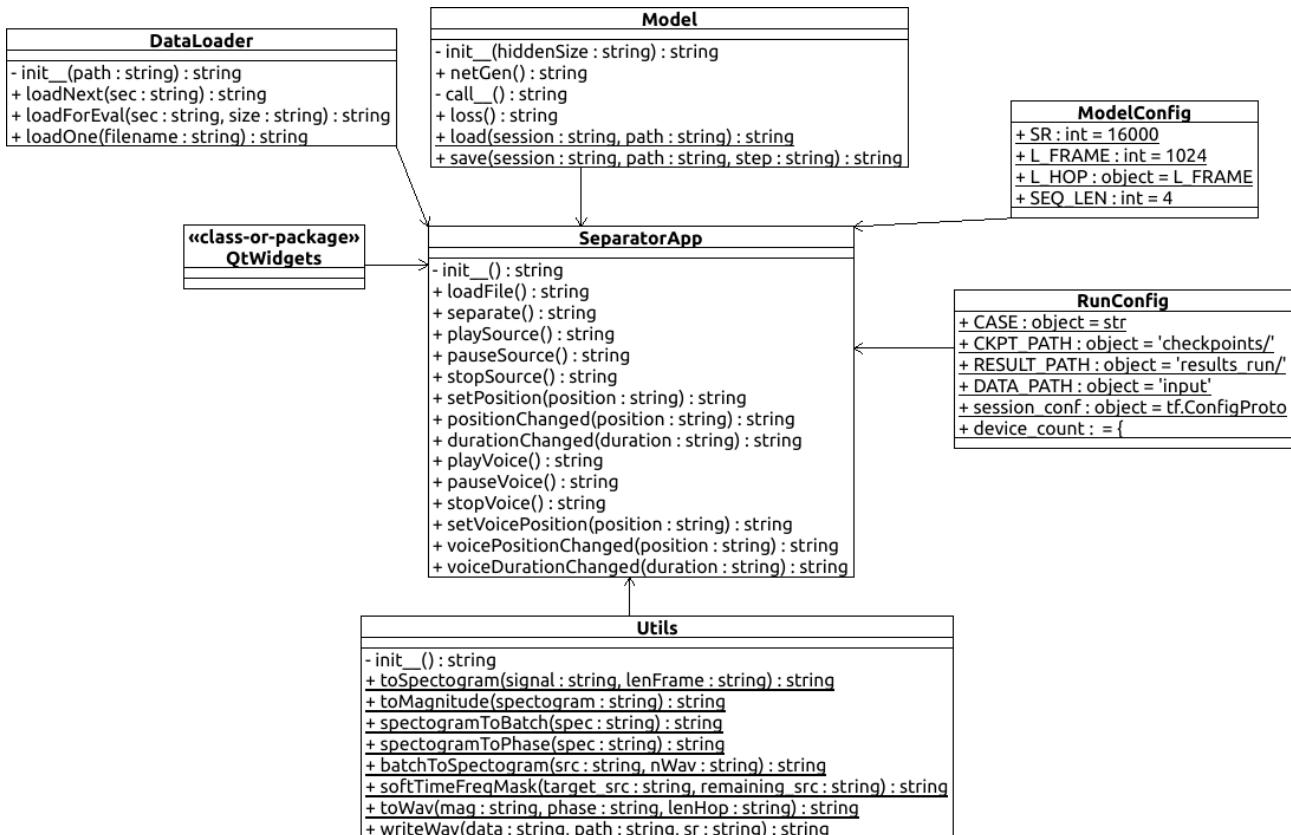


Рисунок 3.2 — Диаграмма классов программы нормального режима

Таблица 3.1 — Описание методов класса DataLoader

Метод	Описание
loadNext(sec)	Считывает очередной wav-файл из тренировочного набора данных, преобразуя его длину в sec
loadForEval(sec, size)	Считывает size случайных wav-файлов из тестового набора данных, преобразуя их длину в sec
loadOne(filename)	Считывает один файл с именем filename

Таблица 3.2 — Описание методов класса Diff

Метод	Описание
update(sec)	Обновление значения ошибки и вычисление разницы с предыдущей ошибкой

## Model

Представляет модель нейронной сети. Описание методов представлено в таблице 3.3.

## ModelConfig

Представляет класс конфигурации для класса Model. Описание атрибутов представлено в таблице 3.4.

## RunConfig

Представляет класс конфигурации для класса SeparationApp. Описание атрибутов представлено в таблице 3.5.

Таблица 3.3 — Описание методов класса Model

Метод	Описание
load(session, path)	Загружает информацию об обученной модели для сессии sess из файлов сохранения, расположенных в path
save(session, path)	Сохраняет информацию о модели из сессии sess в папку path
netGen()	Генерация нейронной сети

Таблица 3.4 — Описание методов класса Model

Атрибут	Описание
SR	Частота дискретизации wav-файла
L_FRAME	Размер окна в ОПФ
L_HOP	Количество аудио кадров между столбцами ОПФ

## SeparationApp

Представляет контроллер работы программы в нормальном режиме. Описание основных методов представлено в таблице 3.6.

### train

Представляет контроллер работы программы в режиме обучения. Описание основных методов представлено в таблице 3.7.

### TrainConfig

Представляет класс конфигурации для класса train. Описание атрибутов представлено в таблице 3.8.

Таблица 3.5 — Описание атрибутов класса RunConfig

Атрибут	Описание
CKPT_PATH	Путь к файлам сохранения обученной модели
RESULT_PATH	Путь сохранения результата работы метода выделения
session_conf	Конфигурация сессии tensorflow

Таблица 3.6 — Описание методов класса SeparationApp

Метод	Описание
loadFile()	Загрузка файла с помощью файлового менеджера
separate()	Исполнение метода выделения источников

## Utils

Представляет вспомогательные функции. Описание методов представлено в таблице 3.9.

### 3.6 Построение нейронной сети

В представлении Tensorflow нейронная сеть является графом потока данных, в котором данные в виде многомерного массива переходят в разные узлы, в процессе чего происходят все необходимые вычисления.

Каждое вычисление в TensorFlow представляется как график потока данных. У него есть два элемента:

- tf.Operation – единицы вычислений.
- tf.Tensor – представляет единицы данных (тензоры).

Таблица 3.7 — Описание методов класса train

Метод	Описание
setup_path()	Настройка путей для файлов сохранения
train()	Обучение нейронной сети

Таблица 3.8 — Описание атрибутов класса TrainConfig

Атрибут	Описание
CKPT_PATH	Путь к файлам сохранения обученной модели
GRAPH_PATH	Путь к файлам сохранения инфографики процесса обучения
DATA_PATH	Путь к файлам обучающей выборки
LR	Коэффициент обучения
MAX_EPOCHS	Количество эпох обучения
session_conf	Конфигурация сессии tensorflow

Алгоритм 2 описывает инициализацию модели. Для определения входных и выходных данных используются тензоры типа `tf.placeholder`. Плейсхолдер нужен исключительно в качестве цели наполнения. Он не инициализирован и не содержит данных.

Таблица 3.9 — Описание методов класса Utils

Метод	Описание
toSpectrogram(signal, lenFrame, lenHop)	Получение спектрограммы сигнала
toMagnitude(spectrogram)	Амплитудная составляющая спектрограммы
spectrogramToPhase(spec)	Фазовая составляющая спектрограммы
toWav(mag, phase, lenHop)	Преобразование спектрограммы в сигнал
writeWav(data, path, sr)	Запись сигнала в аудио файл

```

1 # Input and output shapes
2 inputShape = (None, None, ModelConfig.L_FRAME // 2 + 1)
3 outputMusicShape = (None, None, ModelConfig.L_FRAME // 2 +
1)
4 outputVoiceShape = (None, None, ModelConfig.L_FRAME // 2 +
1)
5 # TF graph input
6 self.input = tf.placeholder(tf.float32, shape=inputShape,
name='mix')
7 # TF graph output
8 self.music = tf.placeholder(tf.float32,
shape=outputMusicShape, name='music')
9 self.voice = tf.placeholder(tf.float32,
shape=outputVoiceShape, name='voice')
10 # Network
11 self.hiddenSize = hiddenSize # Size of hidden layers
12 self.layerCount = nRnnLayer # Count of hidden layers
13 self.network = tf.make_template('network', self.netGen)

```

### Алгоритм 2: Исходный код инициализации модели

Алгоритм 3 описывает генерацию сети. Библиотека Tensorflow позволяет описывать граф многослойной рекуррентной нейронной сети с помощью встроенной функции `tf.nn.dynamic_rnn`. В качестве параметра в функцию передается объект типа `RNNCell`.

Для описания графа обычных слоев используется функция `tf.layers.dense`.

```

1 # Creating of N=layerCount RNN layers
2 rnnLayer = MultiRNNCell([GRUCell(self.hiddenSize) for _ in
3                         range(self.layerCount)])
4 # Creating of RNN network
5 outputRnn, _ = tf.nn.dynamic_rnn(rnnLayer, self.input,
6                                  dtype=tf.float32)
5 inputSize = np.shape(self.input)[2]
6 # Dense layer
7 musicHat = tf.layers.dense(inputs=outputRnn,
8                            units=inputSize, activation=tf.nn.relu, name='musicHat')
8 voiceHat = tf.layers.dense(inputs=outputRnn,
9                            units=inputSize, activation=tf.nn.relu, name='voiceHat')
9 # Time-frequency masking layer
10 retMusic = musicHat / (musicHat + voiceHat +
11                        np.finfo(float).eps) * self.input
11 retVoice = voiceHat / (musicHat + voiceHat +
12                        np.finfo(float).eps) * self.input
12 return retMusic, retVoice

```

**Алгоритм 3:** Исходный код генерации сети  
Итоговый график изображен на рисунке 3.3.

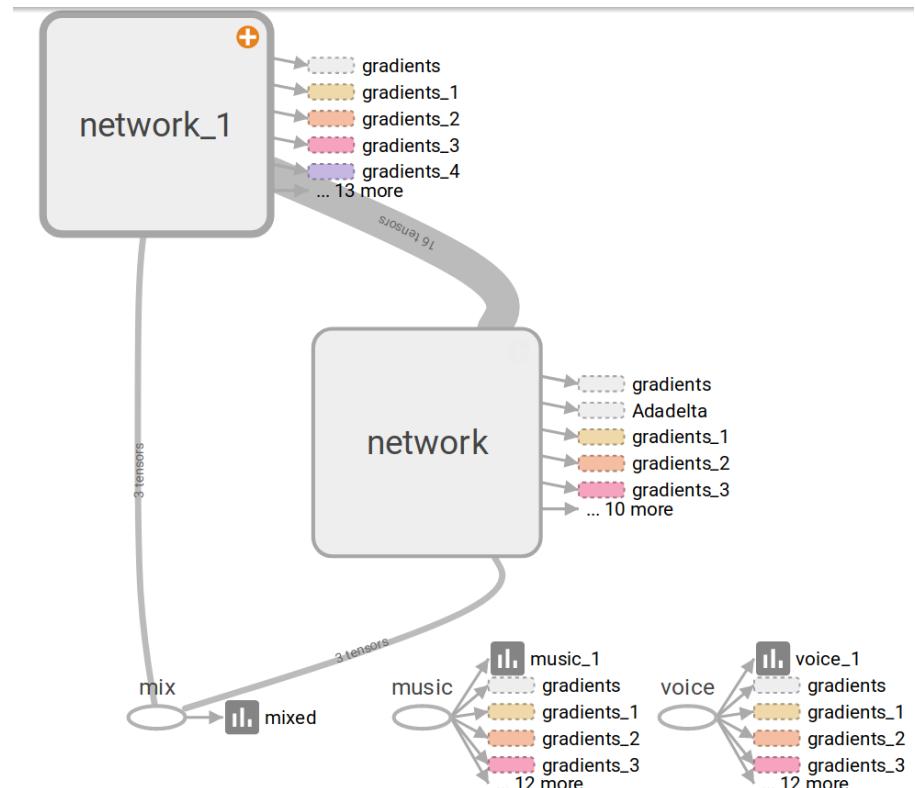


Рисунок 3.3 — Граф модели

## 3.7 Установка программного обеспечения

Для работы программного комплекса необходимо иметь на ПК интерпретатор для Python 3, установщик которого поставляется вместе с программным обеспечением. Для установки необходимых библиотек используется каталог программного обеспечения PyPI (Python Package Index). Все необходимые зависимости указаны в файле requirements.txt, благодаря чему установка библиотек выполняется одной командой:

```
pip3 install -r requirements.txt
```

## 3.8 Руководство пользователя

Разработанный программный комплекс имеет два режима работы: режим обучения и нормальный режим.

### 3.8.1 Режим обучения

Для обучения нейронной сети используется набор данных, путь до которых указывается в поле *DATA\_PATH* класса конфигурации *TrainConfig*. Обучение происходит за *MAX\_EPOCHS* эпох с коэффициентом обучения *LR*. За одну эпоху происходит обработка каждого WAV-файла из набора данных, после которых происходит корректировка весов.

Запуск обучения выполняется командой

```
python3 train.py
```

В процессе обучения генерируются вспомогательные файлы:

- Файлы состояния сети, которые загружаются в последствии для установки обученных весов. Директория для хранения данных файлов задается параметром *CKPT\_PATH*.

- Файлы графиков процесса обучения. Задаются параметром *GRAPH\_PATH*. Для визуализации графиков используется утилита TensorBoard, которая входит в состав библиотеки TensorFlow. Для запуска утилиты используется команда:

```
tensorboard --logdir=/tmp/example/
```

где «/tmp/example/» – путь до сгенерированных файлов графиков. Данная команда запускает локальный сервер, для доступа к которому необходимо открыть в браузере адрес <http://localhost:6006/>.

— Файл журналирования обучения sample.log. В нем отражается информация о работе обучения, в том числе информация об изменении ошибки.

### 3.8.2 Нормальный режим

В нормальном режиме используется уже обученная нейронная сеть. Обученное состояние сети загружается из файлов, полученных во время обучения. Для удобства пользователя программное обеспечение поставляется вместе с файлами. Для запуска программы в нормальном режиме используется команда

**python3 run.py**

После запуска программы появляется главное окно (рисунок 3.4)

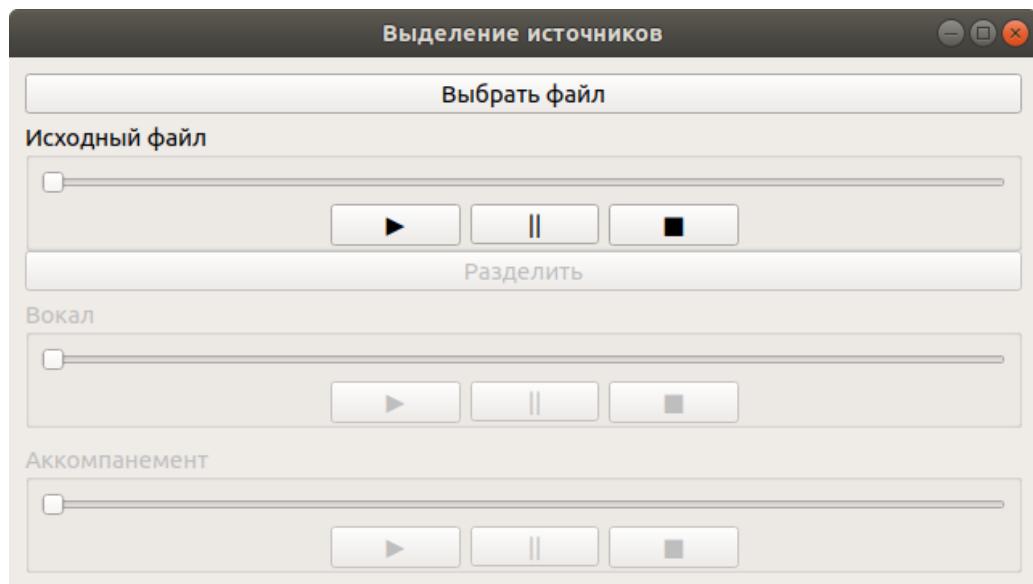


Рисунок 3.4 – Окно программы после запуска

На главном окне представлены следующие элементы:

- a) Кнопка «Выбрать файл». Выводит окно выбора WAV-файла.
- б) Блок «Исходный файл». Управляет воспроизведением исходного аудио файла. Состоит из следующих элементов:
  - 1) Полоса прокрутки. Устанавливает момент воспроизведения, отображает процесс воспроизведения.

- 2) Кнопки «►», «||», «■» (начало воспроизведения, пауза и остановка воспроизведения соответственно). Управляют воспроизведением аудио файла.
- в) Кнопка «Разделить». Инициализирует работу алгоритма выделения. После завершения работы метода активирует блоки «Вокал» и «Аккомпанемент» (рисунок 3.5).
- г) Блок «Вокал». Управляет воспроизведением аудио файла голосовой составляющей. Функционал элементов соответствует блоку «Исходный файл».
- д) Блок «Аккомпанемент». Управляет воспроизведением аудио файла аккомпанемента. Функционал элементов соответствует блоку «Исходный файл».

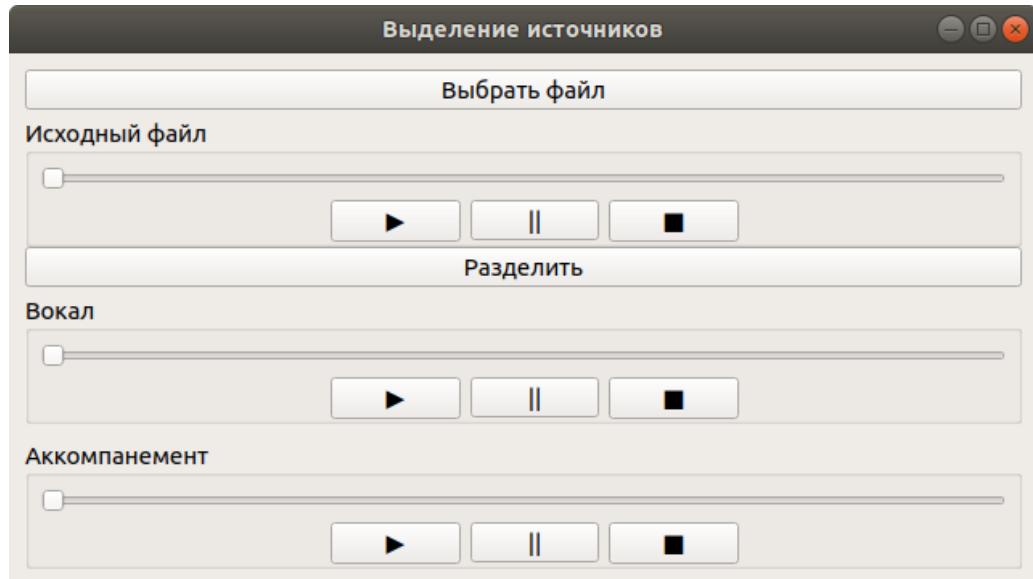


Рисунок 3.5 — Окно программы после работы метода выделения

### 3.9 Вывод

На основе созданной архитектуры был разработан программный продукт на языке Python с использованием библиотек PyQt, TensorFlow, LibROSA и NumPy в среде PyCharm. Продукт был разработан на ЭВМ со следующими характеристиками:

- Процессор: Intel Core i5.
- Частота процессора: 2.80GHz.

- Объем оперативной памяти: 8 Гб.
- Видеокарта: NVIDIA GeForce 840M.
- Операционная система: ubuntu 18.04 LTS.

Обучение нейронной сети происходило на удаленном ЭВМ со следующими характеристиками:

- Количество процессоров: 2.
- Частота каждого из процессоров: 3.20GHz.
- Объем оперативной памяти: 2 Гб.
- Операционная система: ubuntu 16.04 LTS.

## 4 Экспериментальный раздел

В рамках дипломного проекта был проведен ряд экспериментов, направленных на исследование метода выделения голосовой составляющей из монофонического аудио сигнала.

В качестве входных данных использовались музыкальные композиции с мужским и женским вокалом. Так же аккомпанемент мог быть записан либо с использованием музыкальных инструментов, либо при помощи цифровой звуковой рабочей станции.

### 4.1 Описание используемой модели

Обучение модели было произведено один раз за 150 эпох. Данный процесс занял 20 дней. Для обучения использовалась обучающая часть набора данных DSD100. График изменения ошибки в процессе обучения представлен на рисунке 4.1.

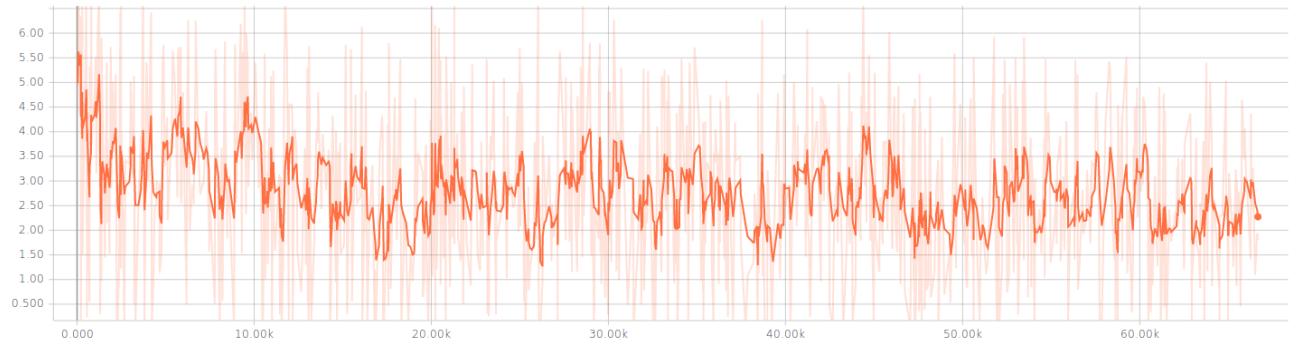


Рисунок 4.1 — График изменения ошибки

### 4.2 Описание тестирующей выборки

Набор данных DSD100 состоит как из данных для обучения нейронной сети, так и из данных для ее проверки. Данные из этого набора были приведены к двухканальному wav-файлу со следующей структурой:

- Левый канал – аккомпанемент.
- Правый канал – голосовая составляющая.

### 4.3 Методика проведения исследования

Проведение эксперимента происходило на всей тестовой выборке набора данных DSD100. Каждый аудио файл проходил обработку разработанным методом и для полученных аудио сигналов голосовой составляющей и аккомпанемента вычислялись метрики SDR, SIR, SAR и ISR, описанные выше.

Среди полученных метрик находились максимальные и минимальные значения. На основании этих чисел определялись средние значения метрик и их разброс.

Сравнение полученных результатов с результатами методов, описанных ранее, представлено в таблице 4.1.

Таблица 4.1 — Сравнение алгоритмов выделения источников

Метод	Метрика	Вокал	Аккомпанемент
РНС	SDR	$-0.6 \pm 4.9$	$4.5 \pm 1.2$
	SIR	$1.9 \pm 6.2$	$14.9 \pm 5.6$
	SAR	$3.6 \pm 2.1$	$16.4 \pm 2.9$
	ISR	$7.3 \pm 2.7$	$6.9 \pm 1.0$
ГНС	SDR	$-8.9 \pm 4.4$	$4.1 \pm 2.6$
	SIR	$10.8 \pm 3.1$	$10.7 \pm 3.8$
	SAR	$-6.6 \pm 4.6$	$12.1 \pm 3.9$
	ISR	$4.8 \pm 2.8$	$5.0 \pm 3.0$
FASST	SDR	$-1.8 \pm 5.8$	$8.9 \pm 3.2$
	SIR	$-3.7 \pm 7.5$	$13.7 \pm 3.0$
	SAR	$4.3 \pm 1.4$	$13.1 \pm 3.2$
	ISR	$5.6 \pm 2.5$	$13.5 \pm 2.5$
Глубинные РНС	SDR	$-5.6 \pm 1.2$	$7.3 \pm 4.2$
	SIR	$-3.6 \pm 2.5$	$10.5 \pm 1.8$
	SAR	$3.8 \pm 1.4$	$10.4 \pm 3.2$
	ISR	$5.3 \pm 2.5$	$5.1 \pm 2.5$

#### **4.4 Спектрограммы**

Были проведены сравнения спектрограмм полученных аудио сигналов с ожидаемыми. Сравнения проводились для двух музыкальных произведений: с мужским вокалом и женским вокалом.

Спектрограммы сигналов голосовой составляющей и аккомпанемента музыкального произведения с мужским вокалом представлены на рисунках 4.2 и 4.3 соответственно.

Спектрограммы сигналов голосовой составляющей и аккомпанемента музыкального произведения с женским вокалом представлены на рисунках 4.4 и 4.5 соответственно.

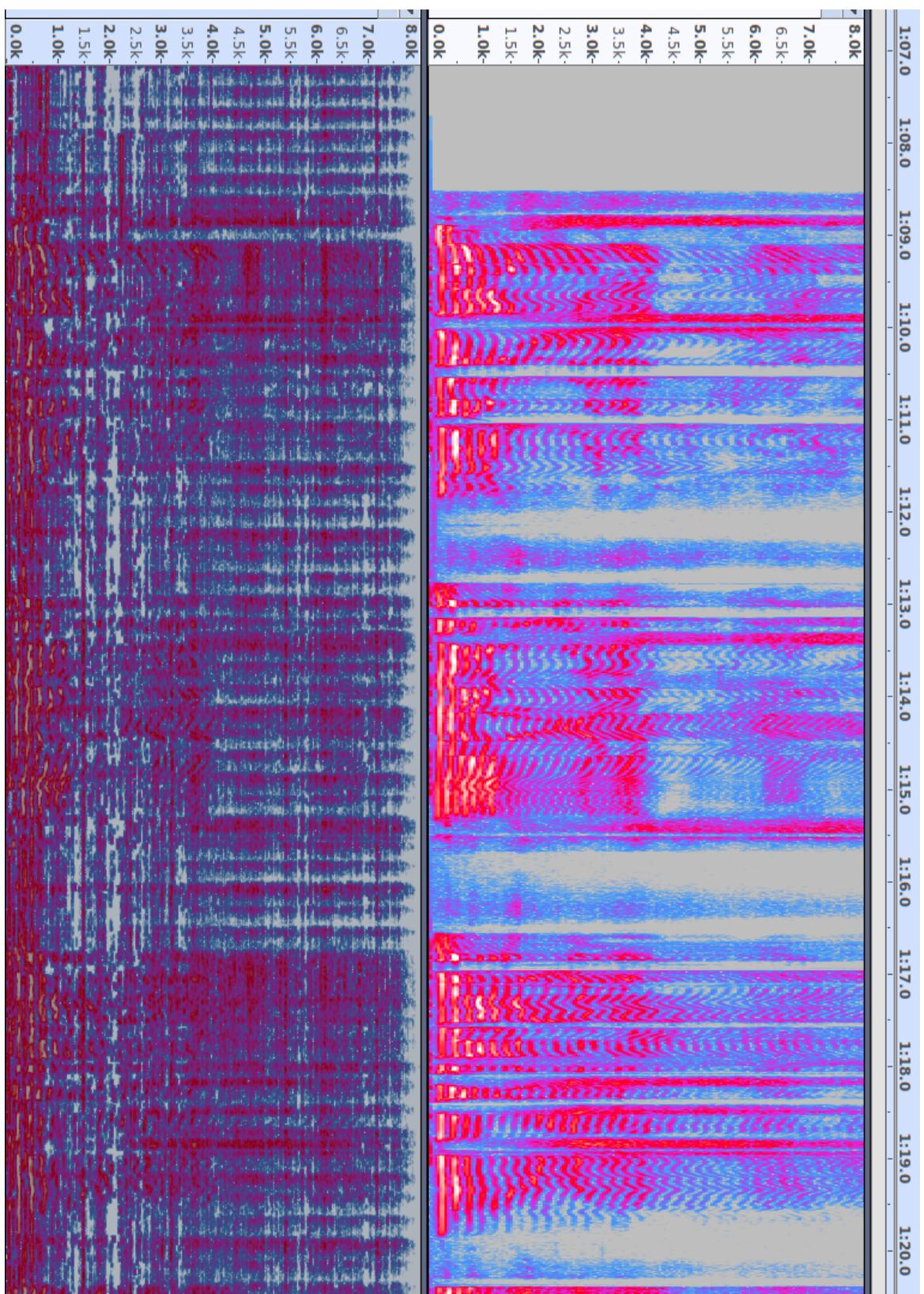


Рисунок 4.2 — Сравнение спектрограмм голосовых составляющих музыкального произведения с мужским вокалом. Сверху: спектрограмма ожидаемого сигнала, снизу: спектрограмма полученного сигнала

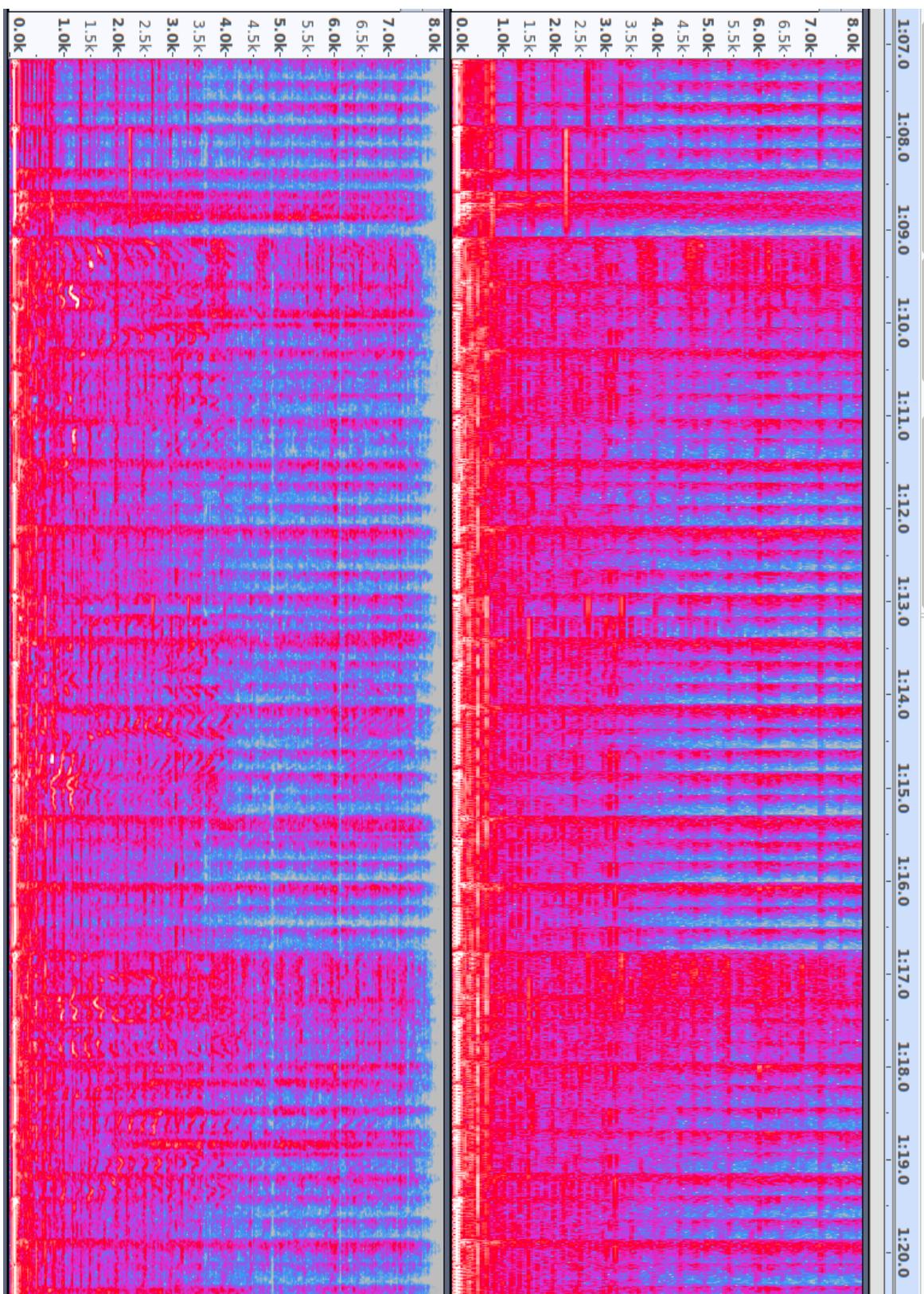


Рисунок 4.3 — Сравнение спектрограмм аккомпанемента музыкального произведения с мужским вокалом. Сверху: спектрограмма ожидаемого сигнала, снизу: спектрограмма полученного сигнала

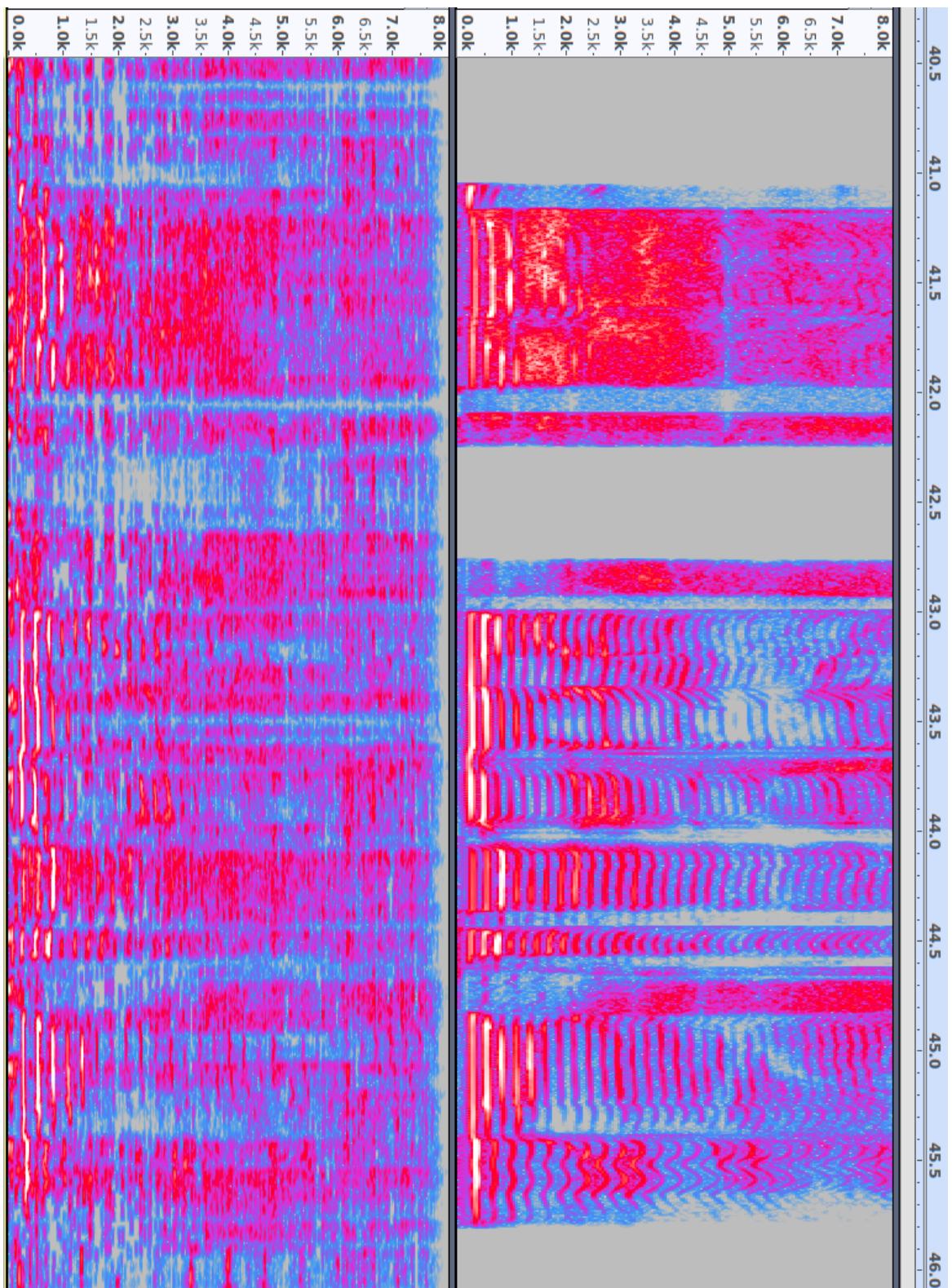


Рисунок 4.4 — Сравнение спектрограмм голосовых составляющих музыкального произведения с женским вокалом. Сверху: спектрограмма ожидаемого сигнала, снизу: спектрограмма полученного сигнала

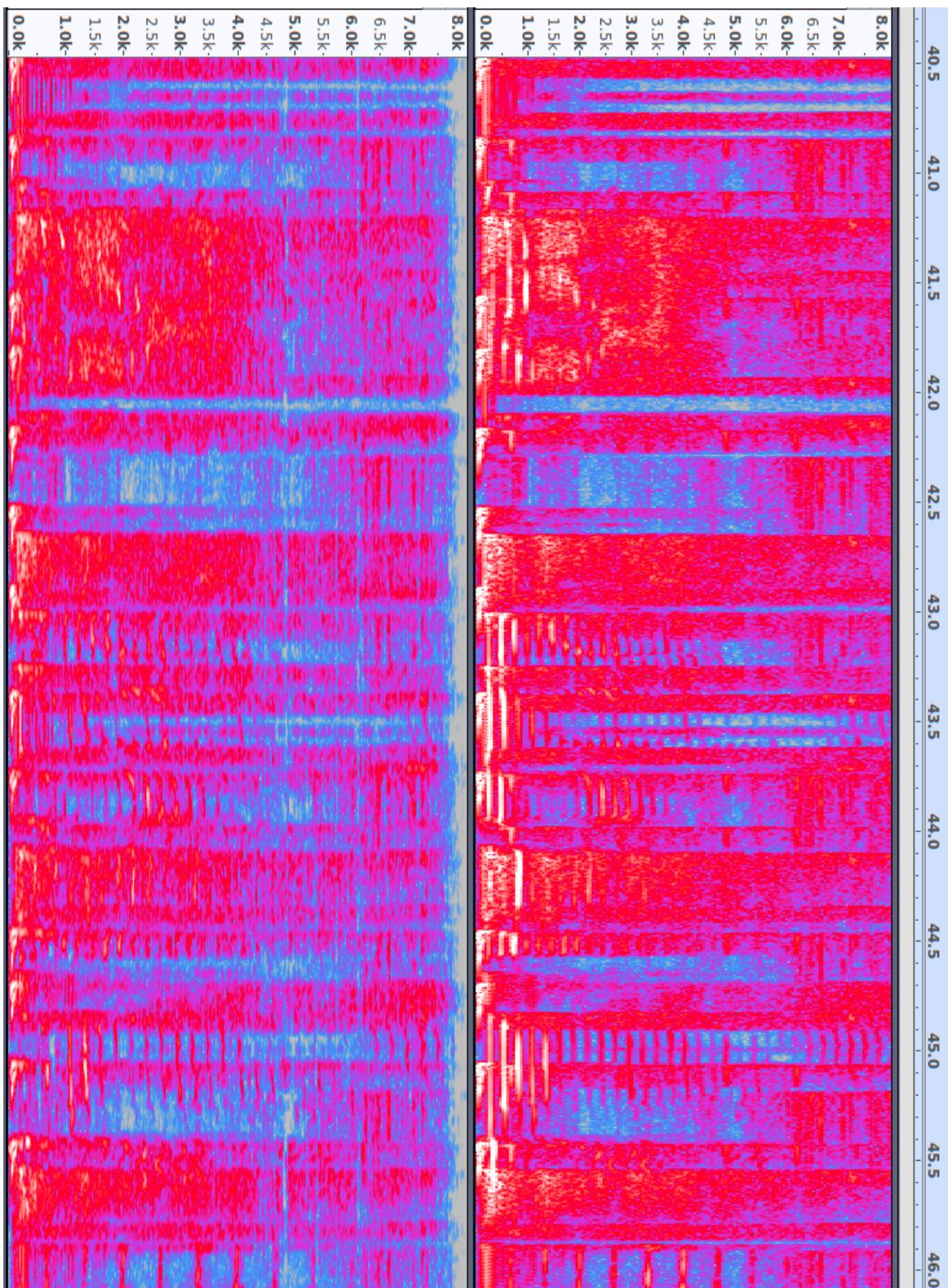


Рисунок 4.5 — Сравнение спектрограмм аккомпанемента музыкального произведения с женским вокалом. Сверху: спектрограмма ожидаемого сигнала, снизу: спектрограмма полученного сигнала

## Заключение

В результате проделанной работы были решены следующие задачи:

- был проведен анализ предметной области;
- были проанализированы существующие решения;
- на основе полученных во время анализа данных был разработан собственный метод выделения голосовой составляющей из монофонического аудио сигнала;
- предложенный метод был реализован в программном продукте.

В результате тестирования и эксперимента было установлено, что разработанный метод:

- для вокала имеет примерно те же показатели метрик, что и метод FASST, при этом имея в среднем в два раза меньший разброс, для аккомпанемента же средние значения метрик в среднем на 60% лучше, имея примерно те же значения разброса;
- для вокала значение метрик в среднем на 80% лучше, чем у метода ГНС, при этом дисперсия в среднем в два раза меньше. Для аккомпанемента значение средних метрик приблизительно одинаковы, но значение дисперсии у разработанного метода на 30% меньше;
- для вокала значение метрик в среднем в 2,5 раза уступают методу СНС, при этом дисперсия в среднем в 2 раза лучше. Для аккомпанемента метрики примерно одинаковые, но разброс у разработанного метода в среднем в 2 раза больше, чем у метода СНС.

В результате тестирования и эксплуатации разработанного ПО замечен основной недостаток – наличие примесей из соседних источников в выделяемом сигнале

Развитие разработанного метода можно осуществлять по следующим направлениям:

- увеличение качества выделения переработкой архитектуры сети;
- определение источников, участвовавших в записи исходного сигнала.

## Список использованных источников

1. *Klapuri, A.* Signal Processing Methods for Music Transcription / A. Klapuri, M. Davy // *Tampere*. — 2004.
2. *Downie, J. S.* Music Information Retrieval / J. S. Downie // *Annual Review of Information Science and Technology*. — 2003.
3. Recurrent Neural Networks for Noise Reduction in Robust ASR / Andrew L. Maas, Quoc V. Le, Tyler M. O’Neil et al. // *INTERSPEECH*. — 2012.
4. Singing-Voice Separation From Monaural Recordings Using Deep Recurrent Neural Networks / Po-Sen Huang, Minje Kim, Mark Hasegawa-Johnson, Paris Smaragdis // *Ismir*. — 2014.
5. *Hsu, C.-L.* On the improvement of singing voice separation for monaural recordings using the MIR-1K dataset. / C.-L. Hsu, J.-S.R. Jang // *IEEE Transactions on Audio, Speech, and Language Processing*. — 2010.
6. *Chandna, Pritish.* Audio Source Separation Using Deep Neural Networks / Pritish Chandna // *TESI DOCTORAL UPF*. — 2016.
7. *Биккенин, Р. Р.* Теория электрической связи / Р. Р. Биккенин, М. Н. Чесноков. — Издательский центр «Академия», 2010. — Р. 336.
8. Multiple channel audio data and WAVE files. [https://docs.microsoft.com/en-us/previous-versions/windows/hardware/design/dn653308\(v=vs.85\)](https://docs.microsoft.com/en-us/previous-versions/windows/hardware/design/dn653308(v=vs.85)).
9. *Mansour, Ali.* Blind Separation of Sources: Methods, Assumptions and Applications / Ali Mansour, Barros Alan Kardec, Ohnishi Noboru // *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*. — 2000.
10. Blind Source Separation of audio signals using independent component analysis and wavelets / P. Lopez, H. Molina Lozano, F. Sanchez, L. N. Oliviera Moreno // *21st International Conference on Electrical Communications and Computers*. — 2011.
11. *Dadula, C. P.* A genetic algorithm for blind source separation based on independent component analysis / C. P. Dadula, E. P Dadios // *International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management*. — 2014.

12. *Miron, Marius.* Improving Score-Informed Source Separation for Classical Music Through Note Refinement / Marius Miron, Julio José Carabias-Ortí Carabias-Ortí, Jordi Janer // *16th International Society for Music Information Retrieval (ISMIR) Conference*. — 2015.
13. *Lee, D. D.* Algorithms for Non-negative Matrix Factorization / D. D. Lee, Seung H. S. // *Advances in Neural Information Processing Systems*. — 2001.
14. *Ozerov, Alexey.* A General Flexible Framework for the Handling of Prior Information in Audio Source Separation / Alexey Ozerov, Emmanuel Vincent, Frédéric Bimbot // *IEEE Transactions on Audio, Speech and Language Processing*. — 2012.
15. *Krizhevsky, Alexey.* ImageNet Classification with Deep Convolutional Neural Networks / Alexey Krizhevsky, Ilya Sutskever, Geoffrey E. Bimbot // *Advances in Neural Information Processing Systems*. — 2012.
16. *Grais, Emad M.* Deep neural networks for single channel source separation / Emad M. Grais, Mehmet Umut Sen, Hakan Erdogan // *Speech and Signal Processing*. — 2014.
17. *Grossberg, Stephen.* Nonlinear neural networks: Principle, mechanisms, and architectures / Stephen Grossberg. — 1988. — 12. — Vol. 1. — Pp. 17–61.
18. *Uhlich, Stefan.* Deep neural network based instrument extraction from music / Stefan Uhlich, Franck Giron, Yuki Mitsufuji // *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*. — 2015.
19. *Hubel, D. H.* Receptive fields and functional architecture of monkey striate cortex / D. H. Hubel, T. N. Wiesel. — 1968.
20. Performance measurement in blind audio source separation / Emmanuel Vincent, Rémi Gribonval, C. Févotte, C Févotte // *IEEE Transactions on Audio, Speech and Language Processing*. — 2006.
21. Professionally-produced music recordings. <https://sisec.inria.fr/sisec-2016/2016-professionally-produced-music-recordings/>.