Министерство науки и высшего образования Российской Федерации
Санкт-Петербургский политехнический университет Петра Великого
Высшая школа интеллектуальных систем и суперкомпьютерных технологий

Работа допущена к защите

Директор ВШИСиСТ ИКНТ

_____ В.М. Ицыксон

«____» _____ 20__ г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

ПРИМЕНЕНИЕ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ В ЗАДАЧЕ СЛЕПОГО РАЗДЕЛЕНИЯ ИСТОЧНИКОВ ЗВУКА В МОНОФОНИЧЕСКОМ МУЗЫКАЛЬНОМ АУДИОСИГНАЛЕ

по направлению подготовки: 02.04.03 Математическое обеспечение и администрирование информационных систем
Направленность (профиль) 02.04.03_02 Проектирование и разработка информационных систем

Выполнил
студент гр. в3540203/70277 А.М. Ивасик

Руководитель
доцент, к.ф.-м.н. В.Г. Пак

Консультант
по нормоконтролю Ю.Д. Заковряшин

Санкт-Петербург
2020

РЕФЕРАТ

На 68 с., 19 рисунков, 46 источников, 3 приложения.

СЛЕПОЕ РАЗДЕЛЕНИЕ ИСТОЧНИКОВ, СЛЕПОЕ РАЗДЕЛЕНИЕ ИСТОЧНИКОВ ЗВУКА, ИСКУССТВЕННЫЕ НЕЙРОННЫЕ СЕТИ, СВЁРТОЧНЫЕ НЕЙРОННЫЕ СЕТИ, АУДИОСИГНАЛЫ, МУЗЫКА

Объектом исследования являются искусственные нейронные сети. Предметом исследования – их применение в задаче слепого разделения источников звука в монофоническом аудиосигнале.

Цель работы – разработка алгоритма слепого разделения источников звука в монофоническом музыкальном аудиосигнале с использованием искусственных нейронных сетей.

На основе экспериментального исследования разработана архитектура свёрточной нейронной сети для разделения источников звука. Осуществлена реализация разработанной архитектуры. Проведена оценка эффективности разработанного алгоритма.

THE ABSTRACT

68 pages, 19 pictures, 46 sources, 3 applications

BLIND SOURCE SEPARATION, BLIND AUDIO SOURCE SEPARATION, ARTIFICIAL NEURAL NETWORKS, CONVOLUTIONAL NEURAL NETWORK, AUDIO SIGNALS, MUSIC

The object of the study are artificial neural networks. The subject of the study is an application of them for blind audio source separation of monophonic musical audio signal.

Objective – development of an algorithm for blind audio source separation of a monophonic musical audio signal based on artificial neural networks.

Based on an experimental study, a convolutional neural network architecture was developed to separate sound sources. The developed architecture was implemented. The effectiveness of the developed algorithm has been evaluated.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Как известно, подавляющее большинство музыкальных произведений являются многоголосными, то есть состоят из звуков, создаваемых различными источниками. При одновременном звучании нескольких источников происходит наложение, в результате которого в воздухе возникают звуковые колебания, представляющие собой суперпозицию множества синусоидальных волн разных амплитуд и частот. Это ставит слуховую систему человека перед проблемой разделения источников звука или, иными словами, выделения отдельных сигналов из смеси. Слуховая система способна успешно группировать различные части звука таким образом, чтобы человек слышал реальные звуки, производимые источниками, а не просто шум.

Разделение источников звука, так легко производимое мозгом человека, является достаточно сложной задачей для компьютера. Формально данная проблема носит название «слепое разделение источников звука» (blind audio source separation) [20, 43], а в сфере звукорежиссуры она известна как «демикширование» (demixing, unmixing). Термин «слепое» в данном случае означает, что разделение происходит без помощи информации (или с очень небольшим количеством информации) о специфических свойствах конкретных сигналов и процессе их смешивания.

В настоящее время актуальной является задача изоляции звуков от отдельных инструментов (в том числе вокала) звучащих в музыкальной композиции. Данная задача является наиболее сложным приложением слепого разделения источников звука [20]. Среди областей применения соответствующей технологии можно выделить следующие:

— Ремастеринг аудиозаписей, в том числе преобразование монофонических записей в стереофонические (upmixing) [10];
— Постпродакшн аудиозаписей;
— Получение минусовых фонограмм (караоке) для развлекательных и образовательных целей;
— Дубляж и реставрация старых кинофильмов;

— Создание электронной музыки;
— Автоматическая транскрипция музыки;
— Распознавание текстов песен;
— Автоматическая генерация субтитров;
— Визуализация музыки.

Стоит отметить, что методы, используемые для разделения источников в музыкальном аудиосигнале, также могут быть применены для улучшения работы слуховых аппаратов [46].

Слепое разделение источников звука является относительно новой технологией цифровой обработки сигналов, первые исследования датируются началом 90-х годов [20]. Однако, на сегодняшний день удалось добиться значительных достижений в этой области. Так, если говорить о задачах, ориентированных на качество звука, то проблема получения минусовых фонограмм в настоящее время считается решённой. При этом проблема выделения сигналов от отдельных музыкальных инструментов или вокала представляет возможности для улучшения. [44]

Среди методов слепого разделения источников звука в последние годы наилучшие результаты показывают подходы, основанные на искусственных нейронных сетях (ИНС) [44].

Объектом исследования настоящей работы являются искусственные нейронные сети. Предметом исследования – алгоритм слепого разделения источников звука в монофоническом аудиосигнале на основе ИНС.

Целью настоящей работы является разработка алгоритма слепого разделения источников звука в монофоническом музыкальном аудиосигнале с использованием искусственных нейронных сетей.

Для достижения поставленной цели определены следующие задачи:
— Обзор методов разделения источников звука, в частности в музыкальном аудиосигнале;
— Обзор архитектур ИНС, применяющихся в задачах разделения источников звука;

— Разработка архитектуры ИНС, описание и анализ алгоритма обучения;
— Подготовка обучающего, валидационного и тестового наборов;
— Программная реализация сети и её тестирование;
— Оценка эффективности по объективным критериям.

ГЛАВА 1. ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Слепое разделение источников

Слепое разделение источников звука (blind audio source separation) является частным случаем слепого разделения источников[1] (blind source separation), которое в свою очередь является технологией цифровой обработки сигналов и заключается в восстановлении сигнала от одного или нескольких источников из имеющейся смеси.

Методы слепого разделения источников применяются в самых разных сферах: обработка изображений [7], сейсмический мониторинг, обработка биомедицинских сигналов (электроэнцефалография, магнитоэнцефалография, электрокардиография) [5]. Применительно к аудиосигналам слепое разделение источников было впервые использовано в сфере коммуникаций в задачах шумопонижения и улучшения речи.

Классической задачей, иллюстрирующей проблематику слепого разделения источников, является проблема коктейльной вечеринки (the cocktail party problem), впервые описанная Колином Черри в 1953 году [8]. Она заключается в распознавании речи одного человека на фоне речи других людей, говорящих в то же время.

1.2. Виды разделения источников звука

В настоящее время выделяют различные виды задач разделения источников звука. В данном разделе приводится описание существующих видов разделения, при этом в процессе изложения производится конкретизация задачи, решаемой в рамках настоящей работы.

В зависимости от полноты исходной информации выделяют слепое и информированное разделение источников звука. Разделение считается информированным, когда доступна информация о специфических свойствах разделяемых сигналов. Термин «слепое» означает отсутствие подобной информации. Стоит

[1] В некоторых источниках на ряду с термином «слепое разделение источников» используется термин «слепое разделение сигналов» (blind signal separation).

оговориться, что, не имея информации о конкретных источниках, мы можем использовать знания и гипотезы о природе разделяемых сигналов. Согласно сказанному во введении, настоящая работа посвящена слепому разделению источников звука.

По доступности смешанного сигнала разделение источников звука может быть пакетным (batch source separation), когда смешанный сигнал доступен целиком на момент разделения, и в реальном времени (online source separation), когда разделение должно выполняться по мере поступления сигнала [20]. В рамках настоящей работы производится разработка пакетного метода разделения.

По характеру приложения задачи разделения источников звука могут быть ориентированными на качество звука (audio quality oriented) и нацеленными на извлечение информации (significance oriented). Данная типология впервые была предложена Э. Винсентом и соавторами [30]. Настоящая работа посвящена разработке метода разделения, ориентированного на качество звука.

Р.М. Пиньон в работе [20] предложил классификацию методов разделения источников звука в зависимости от времени задержки. Данная классификация очевидным образом перекликается с описанными выше. Так, Пиньон выделяет методы с низкой задержкой (low-latency), применяющиеся для разделения в режиме реального времени и для встроенных приложений, и методы с высокой задержкой (high-latency), ориентированные на качество извлекаемого сигнала.

1.3. Разделение источников звука с точки зрения фильтрации

Процесс разделения источников звука по сути является фильтрацией смешанного сигнала. На рисунке 1.1 приведена типизация методов фильтрации, применяемых в разделении источников звука.

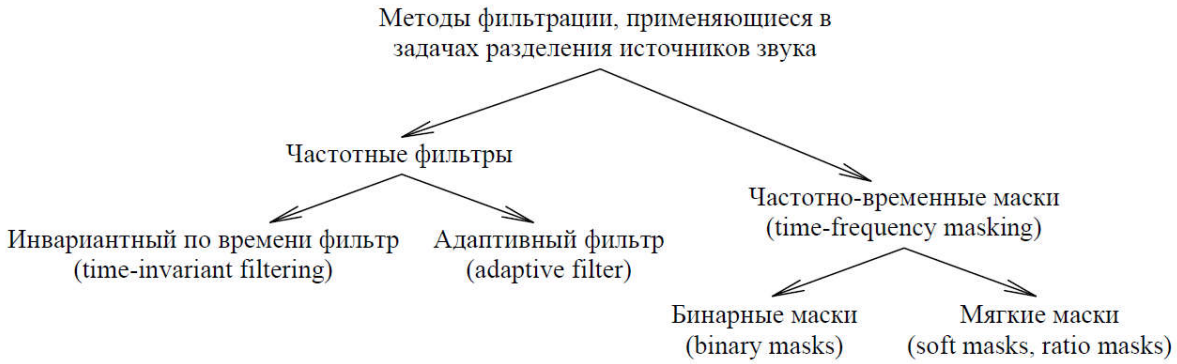

Рис.1.1. Методы фильтрации, применяющиеся при слепом разделении источников звука.

Частотные фильтры представляют традиционный подход к фильтрации сигнала. Использование инвариантных по времени фильтров является наиболее простым способом фильтрации, но не подходит для разделения источников звука в музыкальном аудиосигнале, так как при выделении сигнала, соответствующего определённому источнику, «шум», создаваемый остальными источниками, является изменяющимся во времени. Адаптивные фильтры учитывают динамический характер фонового «шума», но требуют на входе помимо смешанного сигнала сигнал, создающий помехи («шум»). Таким образом, адаптивные фильтры также не могут быть применены в решении поставленной задачи.

Использование частотно-временных масок является наиболее популярным сегодня методом фильтрации аудиосигнала в задачах слепого разделения источников звука. Суть данного метода отражена на рисунке 1.2. Путём оконного преобразования Фурье (short-time Fourier transform) из цифрового аудиосигнала получают спектрограмму[1]. Затем определённым образом, зависящим от конкретного решения, создаётся частотно-временная маска, применение которой к спек-

[1] Спектрограмма – изображение, показывающее зависимость спектральной плотности мощности сигнала от времени. В рамках настоящей работы под спектрограммой понимается двумерная диаграмма, где ось абсцисс представляет время, ось ординат – частоту, третье измерение, представляющее амплитуду на определенной частоте в конкретный момент времени, представляется интенсивностью или цветом каждой точки изображения.

трограмме смешанного сигнала позволяет получить спектрограмму сигнала изолируемого источника. Далее путём обратного преобразования Фурье из спектрограммы отдельного источника получают соответствующий аудиосигнал.

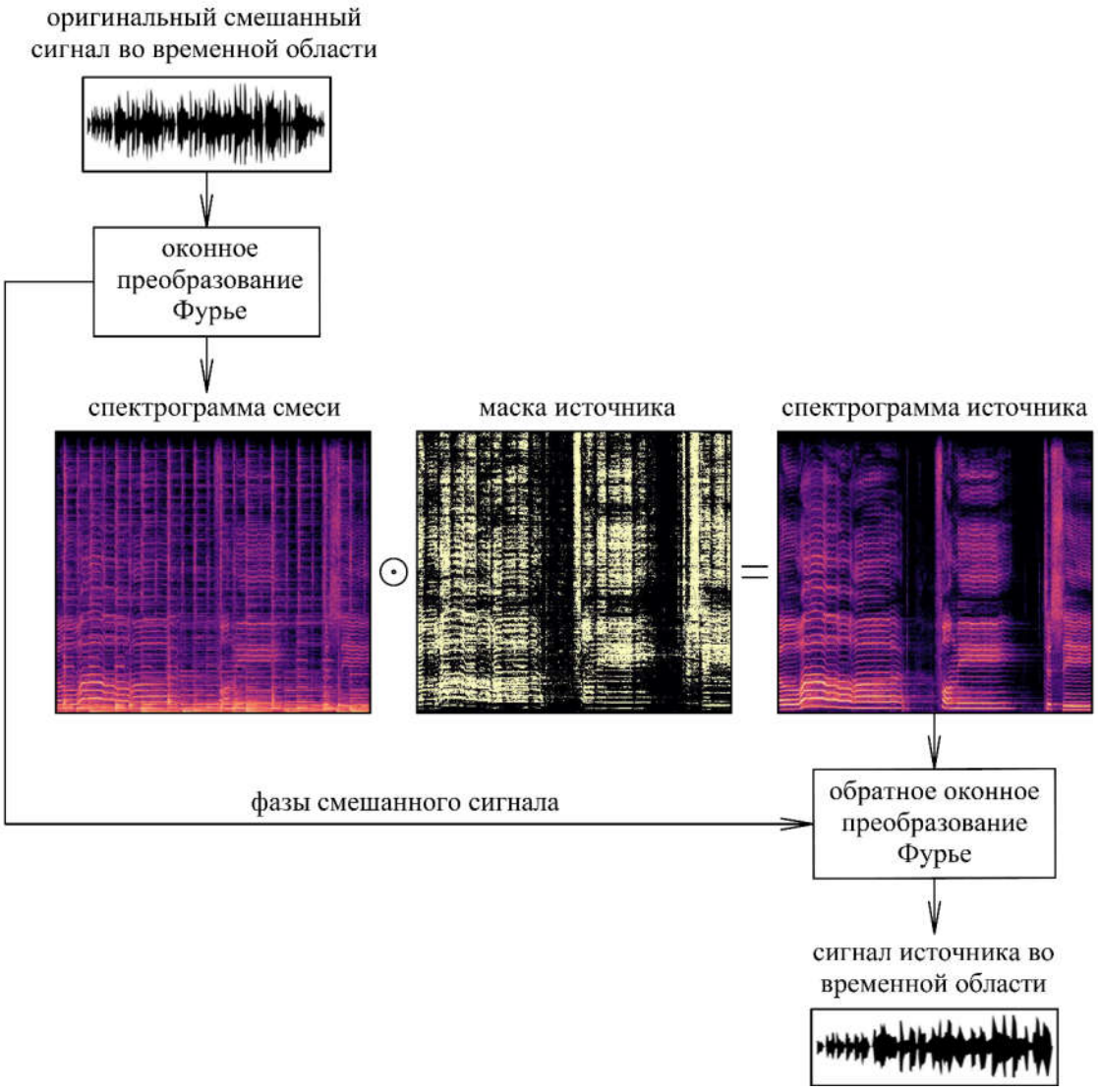

Рис.1.2. Применение бинарной маски для выделения вокала из музыкального аудиосигнала.

В литературе описано два вида частотно- временных масок: бинарные (жесткие) и мягкие. При применении бинарных масок каждая точка (время; частота) спектрограммы смешанного сигнала приписывается строго одному из источников. Важно заметить, что источники звука (инструменты) в музыкальном аудиосигнале имеют пересекающиеся частотные диапазоны и зачастую звучат

одновременно. Таким образом, иногда возникают ситуации наложения, когда амплитуда на определённой частоте в конкретный момент времени складывается из нескольких источников. Данное обстоятельство может сказывается на качестве результатов разделения. Несмотря на это, в силу низкой стоимости вычислений, бинарные маски находят широкое применение, особенно в системах реального времени [20].

Если бинарная маска по сути представляет собой булевскую матрицу, размерность которой равна размерности спектрограммы, то мягкая маска представляет собой матрицу, элементы которой лежат в диапазоне [0; 1]. При использовании мягких масок амплитуды, соответствующие каждой из точек спектрограммы смешанного сигнала, распределяются между разделяемыми источниками пропорционально их вкладу. Позволяя добиться лучшего качества по сравнению с бинарными, мягкие маски увеличивают стоимость вычислений.

При работе с многоканальным аудиосигналом возможны два способа генерации частотно временных масок. В простейшем случае для каждого из каналов независимо от остальных может создаваться своя маска. Иными словами, каждый из каналов обрабатывается как независимый монофонический сигнал. Другой способ заключается в использовании информации о панорамировании[1] источников. Стоит отметить, что использование информации о панорамировании не всегда представляется возможным, так как, во-первых, некоторые музыкальные инструменты традиционно не панорамируются в стереопространстве (вокал, бас, ударные), а во-вторых, большое количество музыкальных записей существует только в формате моно.

Решения, базирующиеся на рассмотренных методах фильтрации, имеют естественный верхний предел качества[2] разделения. В случае, когда помимо смешанного сигнала заранее известны сигналы разделяемых источников, возможно

[1] Панорамирование — это распределение источников звука в стереопространстве. В простейшем случае в процессе панорамирования происходит поиск идеального с точки зрения звукорежиссёра баланса громкостей в каналах для каждого из источников. Так же при панорамировании могут применяться эффекты задержки и динамическая обработка звука.

[2] Метрики, применяемые для оценки качества разбиения описываются в главе 6.

определить так называемые оракулы (oracles), отражающие качественный предел для каждого из типов фильтрации на имеющихся данных. Оперируя оракулами, можно, во-первых, объективно оценить эффективность того или иного алгоритма разделения, а, во-вторых, путём сравнения самих оракулов подобрать стратегию фильтрации наиболее оптимальную для решения поставленной задачи.

Результаты определения оракулов на различных наборах данных описаны Э. Винсентом и др. [28], а также в отчёте Международной кампании по оценке качества разделения источников (Signal Separation Evaluation Campaign (SiSEC)) [44]. Согласно представленным данным наилучшие результаты среди частотно-временных масок показывают мягкие маски, не учитывающие информацию о панорамировании источников.

Таким образом, теоретически лучших результатов можно добиться при использовании мягких масок. Однако, на практике решения, основанные на нейронных сетях, использующие бинарные маски, иногда показывают лучшие результаты по сравнению с аналогичными решениями, использующими мягкие маски [44].

1.4. Степень определённости задач разделения источников

Соотношение между количеством источников и числом смесей (сенсоров, каналов аудиосигнала) является важным и определяющим фактором при выборе метода разделения источников. Возможны три принципиальные ситуации.

Когда количество источников совпадает с количеством исходных смесей, задача разделения является определённой (determined). В случае, когда число источников превышает число смесей, задача является неопределённой (undetermined). И соответственно, если число исходных смесей больше числа разделяемых источников, задача – переопределена (overdetermined).

Если говорить о музыкальном аудиосигнале, то число источников звука не всегда является известным и сильно варьируется, в отличие, например, от задач улучшения речи, где количество источников равно двум: голос и шум. При этом в рамках настоящей работы производится разработка метода разделения

для монофонического аудиосигнала. Таким образом, можно заключить, что рассматриваемая задача является неопределённой.

1.5. Обзор методов разделения источников звука

На сегодняшний день известно множество подходов к разделению источников звука. Их можно разделить на три группы: моделеориентированные; основанные на декомпозиции сигналов и подходы на базе искусственных нейронных сетей. (см. рис. 1.3)

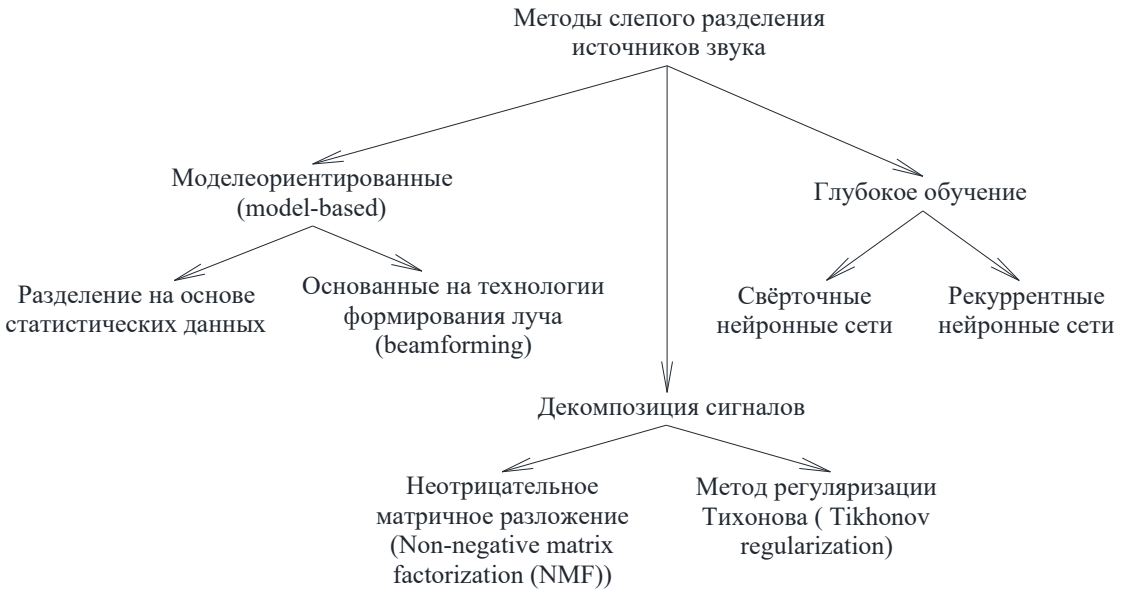

Рис.1.3. Методы разделения источников звука.

1.5.1. Моделеориентированные методы

Моделеориентированные подходы к разделению аудиосигнала представлены в свою очередь двумя группами методов. К методам первой группы относятся декорреляционные техники, а также считающиеся традиционными для слепого разделения источников анализ независимых компонент (independent component analysis) и метод главных компонент (principal component analysis). В основе перечисленных методов лежит использование знаний и гипотез о статистике разделяемых источников.

Вторая группа моделеориентированных методов основана на так называемой технологии формирования луча (beamforming). В их основе лежит использование информации о пространственном положении источников звука и сенсоров (микрофонов). Наглядной иллюстрацией данной группы методов является микрофонная решётка (микрофонный массив) [41].

Важно отметить, что качество результатов разделения рассмотренных методов сильно зависит от принимаемых моделей. Любое несоответствие между разделяемым сигналом и принятой моделью становится источником артефактов и шума. При этом моделеориентированные методы разделения источников звука применимы только для решения определённых и переопределённых задач. Таким образом, описанные моделеориентированные подходы не годятся для реализации в рамках настоящей работы.

1.5.2. Декомпозиция сигналов

Методы, основывающиеся на декомпозиции сигнала, являются более современным подходом к разделению источников звука. Их суть заключается в нахождении компонентов, формирующих исходный смешанный сигнал, и вычислении посредством их группировки сигналов отдельных источников.

До последнего времени наиболее популярные методы разделения источников звука базировались на неотрицательном матричном разложении (NMP) (non-negative matrix factorization). В основе данного подхода лежит предположение о том, что спектр смешанного сигнала может быть смоделирован как линейная комбинация элементарных неотрицательных спектров (базисных компонентов). Известно, что решения на базе NMP показывают удовлетворительные результаты. [43, 11]

В 2013 году Р.М. Пиньон [20] предложил в качестве дополнения к NMP использовать метод регуляризации Тихонова (Tikhonov Regularization). Использование данного подхода позволяет снизить задержку и стоимость вычислений по сравнению с NMP, соответственно отлично подходит для разделения источников звука в реальном времени.

1.5.3. Глубокое обучение

Согласно отчёту Международной кампании по оценке качества разделения источников (Signal Separation Evaluation Campaign (SiSEC)) [44] за последние два года произошёл резкий всплеск интереса к проблеме разделения источников звука, при этом большинство решений, разработанных участниками сообщества основаны на глубоком обучении. Сравнение порядка 30 решений, присланных в 2018 году для участия в кампании SiSEC, показало, что методы, базирующиеся на нейронных сетях, показывают наилучшие результаты. Важно заметить, что некоторые из решений на основе нейронных сетей в 50% случаев показывают результаты сопоставимые с оракулами. Таким образом, глубокие нейронные сети являются очевидным выбором для решения задачи, поставленной в настоящей работе.

Обзор архитектур искусственных нейронных сетей (ИНС), применяющихся в задачах разделения источников звука, приведён в главе 2.

ГЛАВА 2. ОБЗОР АРХИТЕКТУР ИНС, ПРИМЕНЯЮЩИХСЯ В ЗАДАЧАХ РАЗДЕЛЕНИЯ ИСТОЧНИКОВ ЗВУКА

Основные виды архитектур нейронных сетей, включая свёрточные сети, автокодировщики и рекуррентные сети, известны с конца восьмидесятых годов прошлого века. Однако настоящую популярность ИНС обрели лишь в результате революции в машинном обучении середины 2000-х, основными причинами которой послужили работы Джеффри Хинтона [15, 16], Иошуа Бенджио [6], а также прогресс в вычислительной технике и появление наборов данных, имеющих достаточные размеры.

Первый промышленный успех нейронных сетей был связан с применением их в сфере распознавания речи. Сегодня индустриальные приложения на основе глубокого обучения находят применение в самых разных областях: компьютерном зрении, распознавании речи и аудиозаписей, обработке естественных языков, робототехнике, биоинформатике и химии, видео-играх, поисковых системах, интернет-рекламе и финансах. Достижения систем на базе ИНС в перечисленных сферах ПО пробудили интерес к их применению и в области разделения источников звука.

В настоящей главе приводится обзор архитектур нейронных сетей, применяющихся в задачах разделения источников звука. Общий анализ основных архитектур ИНС известных на сегодняшний день широко освещён в книгах И. Гудфеллоу, И. Бенджио, А. Курвилля [2], а также С. Николенко, А. Кадурина, Е.Архангельской [4] и выходит за рамки настоящей работы.

2.1. Полносвязные сети

Полносвязные нейронные сети (dense neural networks, DNN) не применяются для решения задач разделения источников звука в силу неэффективности по сравнению с другими типами архитектур. Дело в том, что полносвязные ИНС не учитывают топологию входных данных. Так, например, при работе с изображениями (спектрограммами аудиосигнала), соседние пиксели подаются на вход

полносвязной сети как независимые компоненты. Это означает, что сеть в процессе обучения должна сперва понять, что некоторые компоненты входного вектора сильно скоррелированы [4].

В работе [21] приведено сравнение полносвязной и свёрточной сетей применительно к задаче разделения источников звука. Авторы отмечают, что свёрточная сеть с меньшим количеством параметров чем у полносвязного аналога показывает лучшие результаты.

2.2. Свёрточные сети

Свёрточные нейронные сети (convolutional neural networks) являются одной из наиболее популярных архитектур ИНС и предназначены для обработки данных с сеточной топологией. Основным приложением, ради которого были придуманы свёрточные сети, является обработка изображений. Путём преобразования Фурье аудиосигнал может быть преобразован в двумерный тензор, строки которого соответствуют частотам, а столбцы – моментам времени. Полученный тензор (спектрограмма) по сути является представлением аудиосигнала в виде изображения и может быть подан на вход свёрточной сети.

На сегодняшний день известно несколько архитектур свёрточных сетей, применяемых для решения задачи разделения источников звука в музыкальном аудиосигнале.

2.2.1. Классические свёрточные сети

Под классическими свёрточними сетями в рамках настоящей работы будем понимать сети из нескольких рядовых свёрточных слоёв, состоящих в общем случае из блоков свёртки, нелинейной функции активации и пулинга[1] (pooling); и нескольких полносвязных слоёв на выходе сети. Примером классической архитектуры свёрточной сети является известная архитектура VGG [24].

В качестве примера ИНС с классической архитектурой, применяемых для разделения источников звука, можно привести архитектуру Ж. Рома, О. Грина и П.А. Трамбле [21], а также архитектуру Э. Корецкого [36].

[1] В ряде источников используется термин «субдискретизация».

ИНС Рома, Грина и Трамбле содержит три последовательных свёрточных слоя, состоящих из блоков свёртки с ядром 5×5, активации ReLU и пулинга с ядром 2×2, и завершается двумя полносвязными слоями (см. рис.2.1.). Число карт признаков (каналов) постепенно растет на более глубоких уровнях сети от 4 до 32. На вход сети подаётся отрезок спектрограммы, соответствующий 11 шагам преобразования Фурье (≈130 мс). На выходе – вектор, представляющий собой единичный отрезок искомой мягкой маски. В качестве функции ошибки используется сумма квадратов отклонений (mean squared error), в качестве оптимизатора – адаптивный вариант градиентного спуска ADAM.

К недостаткам описанной архитектуры можно отнести использование свёрточных блоков с ядром 5×5. Известно [4], что два идущих подряд свёрточных блока с ядром 3×3 имеют рецептивное поле размером 5×5, количество весов при этом у них будет 18 против 25. Таким образом, сеть может стать глубже, уменьшая при этом общее количество весов. Наличие дополнительной нелинейности между слоями позволяет увеличить «разрешающую способность» по сравнению с единственным слоем с большей свёрткой.

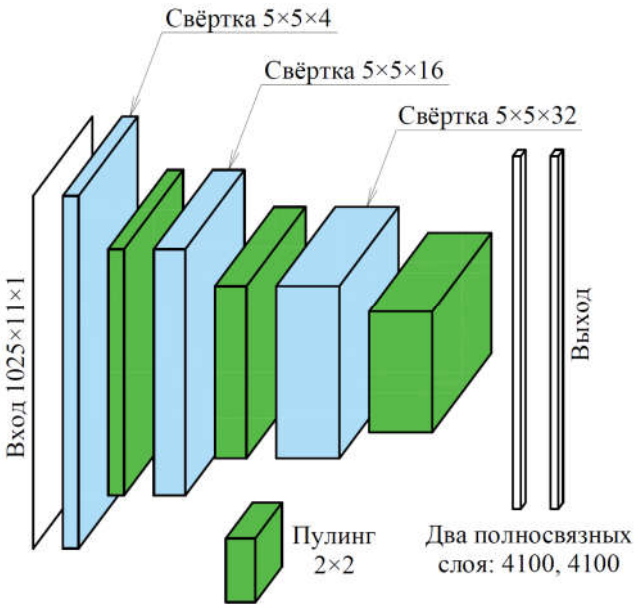

Рис.2.1. Схема архитектуры свёрточной сети Ж. Рома, О. Грином и П.А. Трамбле [21].

ИНС Э. Корецкого содержит четыре свёрточных слоя состоящих из блоков свёртки с ядром 3×3 и активации LeakyReLU. Через каждые два слоя предусмотрен блок пулинга с ядром 3×3. Завершается сеть двумя полносвязными слоями. Число карт признаков (каналов) колеблется от слоя к слою (см. рис.2.2.). На вход сети подаётся отрезок спектрограммы, соответствующий 25 шагам преобразования Фурье (≈290 мс). На выходе – вектор, представляющий собой единичный отрезок искомой бинарной маски. В качестве функции ошибки используется сумма квадратов отклонений, хотя по сути получение бинарной маски скорее является задачей классификации (см.п.3.2). В качестве оптимизатора используется стохастический градиентный спуск.

К недостаткам описанной модели можно отнести использование на выходном слое сети линейной функции активации. Очевидно, что применение функции сигмоида, имеющей область значений (0;1), может облегчить задачу сети при обучении на частотно-временных масках, значения элементов которых находятся в диапазоне [0;1].

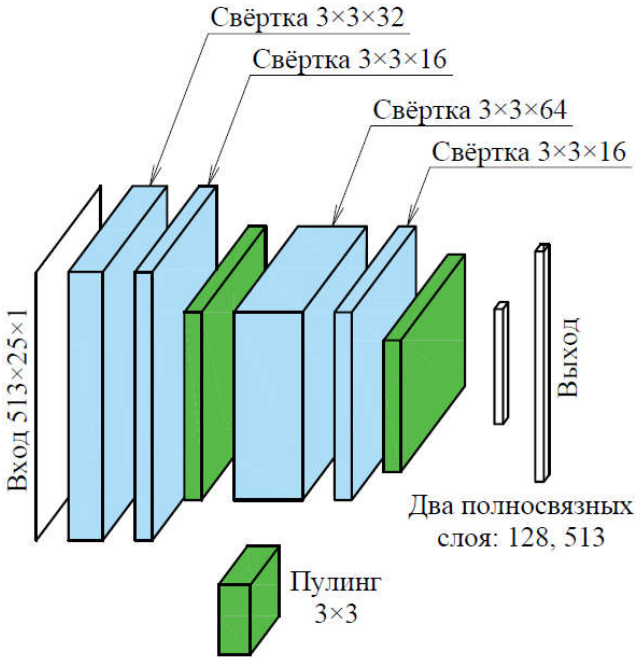

Рис.2.2. Схема архитектуры свёрточной сети Э. Корецкого [36].

2.2.2. U-Net сети

Архитектура U-Net была разработана для сегментации биомедицинских изображений [22] и по сути представляет собой свёрточную архитектуру типа кодировщик-декодировщик. В качестве кодировщика выступает классическая свёрточная сеть, каждый из слоёв которой уменьшает входное изображение, увеличивая при этом количество каналов. Декодировщик представляет собой развёрточную нейронную сеть (deconvolutional neural network) [19], каждый из слоёв которой соответственно увеличивает входное изображение, уменьшая количество каналов. Важно отметить, что архитектура U-Net предусматривает соединения соответствующих слоёв кодировщика и декодировщика. Исходное изображение, проходя через сеть, сжимается, минует узкое место на стыке кодировщика и декодировщика и разжимается до первоначального размера.

Сегментация изображения – это процесс его разбиения на сегменты путём присвоения меток каждому пикселю. Создание маски для источника звука на основе спектрограммы смешанного сигнала по сути является её сегментацией. Данная идея подвигла исследователей к адаптации U-Net сетей для решения задачи разделения источников звука. В качестве примеров отметим две реализации.

Архитектура А. Янссона, Э. Хамфри, Н. Монтеккио, Р. Биттнер, А. Кумар, Т. Вейда [17] представляет собой вариацию классической U-Net архитектуры (см.рис.2.3). Кодировщик и декодировщик имеют глубину в 6 слоёв. Ядро свёртки различных слоёв имеет размер 5×5 и шаг равный 2. В качестве функции ошибки принята сумма модулей отклонений (mean absolute error). Оптимизатор – ADAM.

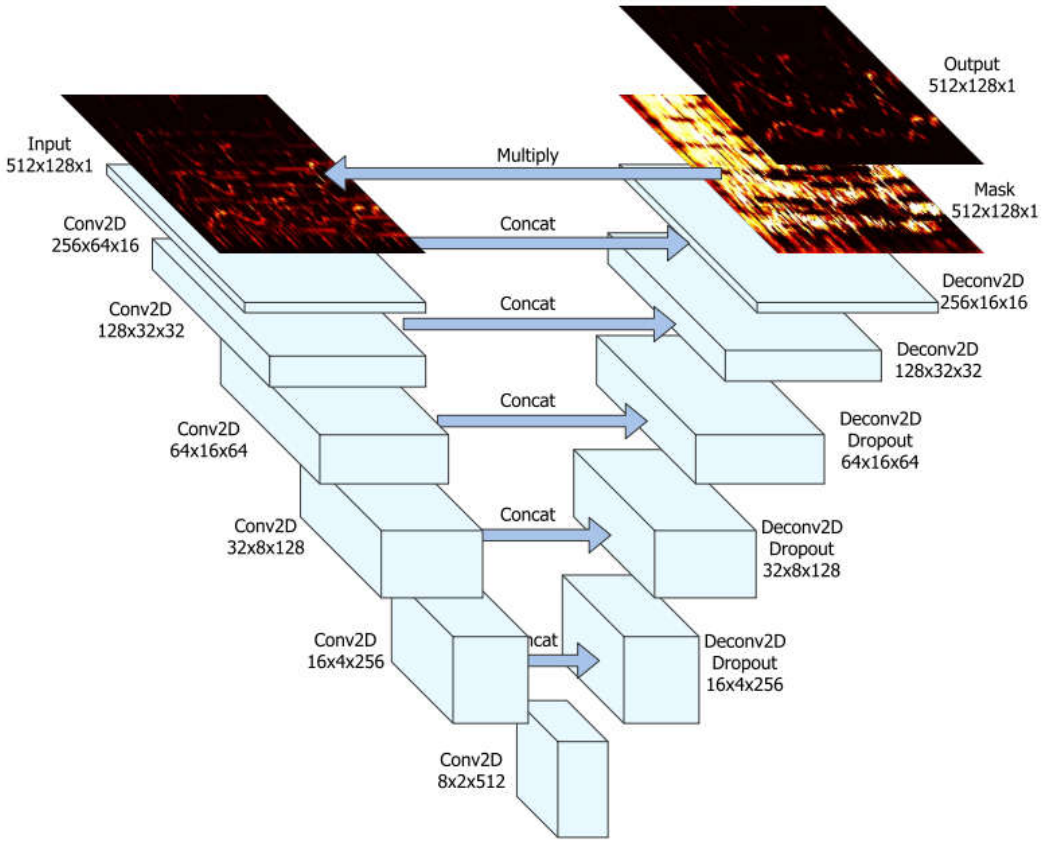

Рис.2.3. Схема архитектуры U-Net предложенная А. Янссоном, Э. Хамфри, Н. Монтеккио, Р. Биттнер, А. Кумар, Т. Вейдом [17].

Архитектура Д. Столлера, С. Эверта, С. Диксона [25] представляет одномерную адаптацию U-Net архитектуры. Авторы называют предложенную ими специфическую архитектуру Wave-U-Net, подчеркивая, что она разработана специально для работы с аудиосигналом. Вместо временного отрезка спектрограммы на вход сети подаётся отрезок формы волны смешанного аудиосигнала, на выходе – соответствующий отрезок формы волны источника.

2.3. Рекуррентные сети

Рекуррентные нейронные сети – это семейство нейронных сетей, предназначенных специально для обработки последовательных данных, к которым, можно отнести аудиосигналы. Архитектуры рекуррентных сетей делятся на три группы: обычные рекуррентные сети, сети с долгой краткосрочной памятью (long short term memory (LSTM)) и вентильные сети (gated recurrent unit (GRU)). Подробный обзор перечисленных видов архитектур приведён в литературе [4, 2]

и выходит за рамки настоящей работы. Ниже приведён обзор известных архитектур рекуррентных сетей, применяемых в задачах распознавания.

2.3.1. Сети с долгой краткосрочной памятью (LSTM)

Основной идеей, лежащей в основе архитектуры рекуррентных сетей является наличие в рекуррентных слоях скрытых состояний, запоминающих историю приходящих данных и использующихся для предсказания будущих элементов последовательности. Как известно, в обычных рекуррентных сетях влияние скрытого состояния на последующие состояния рекуррентной сети экспоненциально затухает. Для преодоления указанной проблемы были предложены сети с долгой краткосрочной памятью, скрытое состояние в которых представляет не единственное число, а специального вида ячейку, в которой явным образом смоделированы «долгая память», процессы записи и чтения из этой «ячейки памяти». [4]

В качестве примера LSTM сети, разработанной для решения задачи разделения источников звука, можно привести архитектуру, предложенную С. Уличем, М. Порку, Ф. Хироном, М. Эненклом, Т. Кемпом, Н. Такахаши и Ю. Мицуфуджи [27], представляющую собой двунаправленную сеть с долгой краткосрочной памятью, состоящую из трёх слоёв, каждый из которых содержит 500 ячеек.

2.3.2. Вентильные сети (GRU)

Основным стимулом для разработки вентильных сетей было желание снизить количество параметров в LSTM сетях, сохранив при этом эффект долгосрочной памяти.

В качестве примера вентильной сети, использующейся для разделения источников звука, можно указать модель, разработанную Д. Уорд и Ц. Ц. Конг [31] и представленную на Кампании по оценке качества методов разделения сигнала 2018 [44]. Данная сеть состоит из трёх двунаправленных GRU слоёв, каждый из которых содержит по 521 ячейке.

2.4. Сети смешанного типа

Иногда на практике применяются довольно экзотические модели ИНС, сочетающие в себе элементы разных видов архитектур. Так, среди моделей ИНС применяющихся для решения разделения источников звука встречаются архитектуры, сочетающие в себе элементы свёрточных U-Net и рекуррентных сетей.

К архитектурам данной группы относится сеть Н. Такахаси, Н. Госвами, Ю. Мицуфудзи [26], К. Дура [9], а также сеть, разработанная С.И. Мимилакисом, К. Дроссоси, Ж.Ф. Сантосом, Д. Шуллером, Т. Виртанени, И. Бенджио [18].

ГЛАВА 3. РАЗРАБОТКА АРХИТЕКТУРЫ ИНС, ОПИСАНИЕ АЛГОРИТМА ОБУЧЕНИЯ

3.1. Формат и содержание обучающих примеров[1]

При проектировании нейронной сети для решения конкретной задачи прежде всего важно определиться с форматом входных и выходных данных. Иными словами, необходимо сформировать чёткое представление о размерах и содержании тензоров, составляющих размеченные примеры.

В качестве входных данных в разработанной модели ИНС используются отрезки спектрограмм смешанного аудиосигнала, значения мощностей которых в пределах отдельных отрезков нормализованы к диапазону [0; 1]. В качестве выходных данных принят вектор, представляющий собой бинарную маску для единичного отрезка спектрограммы аудиосигнала изолируемого источника, соответствующего центру входного отрезка спектрограммы смешанного аудиосигнала (см.рис.3.1.). Значение «1» в определённой точке бинарной маски означает, что вклад изолируемого источника в общую мощность соответствующей частоты в соответствующий момент времени составляет не менее 50%. В противном случае в данной точке маски находится «0».

[1] Материал данного пункта затрагивает дискретное преобразования Фурье, подробное рассмотрение которого выходит за рамки настоящей работы. За детальным изложением сути алгоритма дискретного преобразования Фурье читатель может обратиться к следующим источникам: [32, 14].

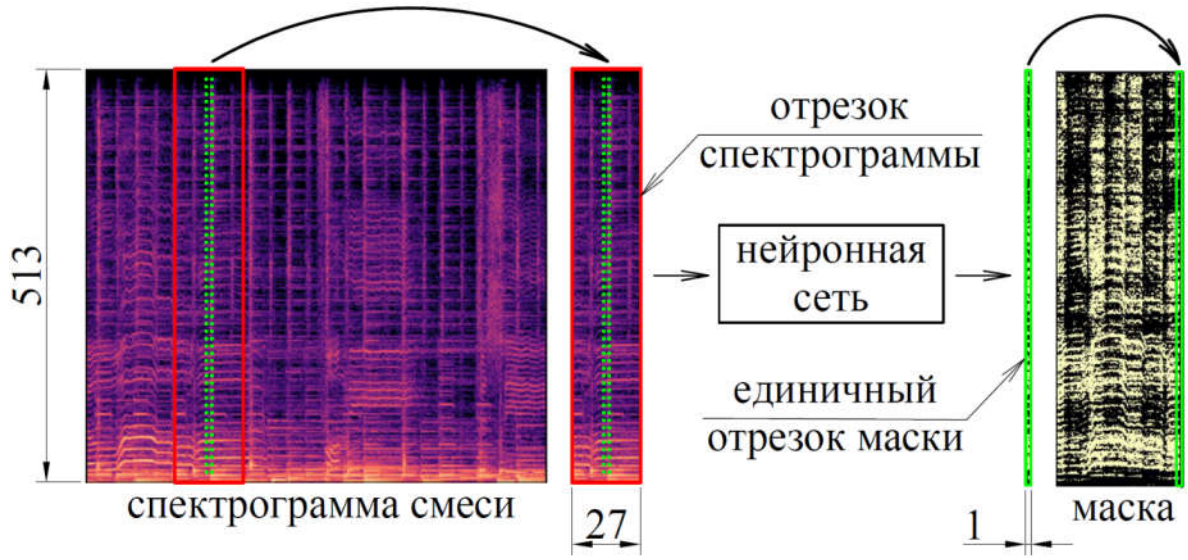

Рис.3.1. Формат входных и выходных данных разрабатываемой нейронной сети.

3.1.1. Высота спектрограммы

Высота спектрограммы соответствует числу диапазонов (frequency bins), на которое разбивается спектр частот аудиосигнала в результате дискретного преобразование Фурье. Три основных параметра, влияющих на это число: частота дискретизации аудиосигнала (sampling rate), размер окна (window size) и шаг (hop).

Частота дискретизации по умолчанию для абсолютного большинства музыкальных аудиосигналов равняется 44,1 (кГц), что позволяет закодировать звук частотой до 22,05 (кГц), что в свою очередь соответствует пределу человеческого слуха. При решении задач разделения источников звука в музыкальном аудиосигнале частота дискретизации исходного сигнала часто понижается с целью уменьшения объёма вычислений (см.прил.1). В рамках настоящей работы разделение источников производится на примере выделения вокала из смешанного музыкального аудиосигнала (см.п.4.2.). Известно [34], что частота самого высокого звука, который может произвести речевой аппарат человека достигает

10 (кГц)[1]. Таким образом, частота дискретизации исходного сигнала может быть снижена до 22,05 (кГц) без потери качества.

Размер окна дискретного преобразования Фурье влияет на частотное и временное разрешение получаемой спектрограммы. Больший размер окна позволяет разбить спектр аудиосигнала на большее количество диапазонов и, следовательно, увеличивает частотное разрешение полученной спектрограммы. Однако при этом снижается её временное разрешение, что может быть критично при обработке музыкальных сигналов. Меньший размер окна уменьшает количество диапазонов, на которые разбивается спектр, что снижает частотное разрешение спектрограммы, увеличивая временное разрешение. Фактически применение окна слишком маленького размера может привести к тому, что две частоты, соответствующие разным басовым нотам, будут отнесены к одному диапазону разбиения спектра и станут неразличимы на спектрограмме.

Шаг дискретного преобразования, принимаемый меньшим размера окна, приводит к увеличению временного разрешения спектрограммы.

В упомянутых ранее (см.гл.2.) решениях применяются различные комбинации значений частоты дискретизации, размера окна и шага (см.прил.1). В настоящей работе с целью уменьшения размера входного тензора и, следовательно, снижения объёма вычислений размер окна принимается равным 1024, а шаг – 256.

3.1.2. Длина отрезка спектрограммы

Как было сказано ранее длина отрезка спектрограммы, представляющего входной тензор, зависит от архитектуры конкретной сети. Так, в решениях на основе классических свёрточных сетей [21] и [36] на вход подаются отрезки спектрограмм длиной 11 и 25 шагов преобразования Фурье соответственно, что равняется 130 (мс) и 290 (мс). В архитектурах типа U-Net применяются отрезки большей длины, в частности, длина спектрограмм, подающихся на вход сети [9], соответствует 128 шагам преобразования Фурье или 1,47 (с).

[1] Данная частота соответствует свистящему согласному звуку «с».

В рамках настоящей работы длина отрезка спектрограммы, подающейся на вход разработанной сети (см.п.3.6.2.), принимается равной **27** шагов преобразования Фурье для возможности применения трёх слоёв пулинга с размером ядра 3×3.

3.2. Функция ошибки

Выбор функции ошибки (функции стоимости) является важнейшим аспектом проектирования нейронной сети. В простейших случаях для задач классификации в качестве функции ошибки применяется перекрёстная энтропия (cross entropy), для задач регрессии – сумма квадратов отклонений (mean squared error) или сумма модулей отклонений (mean absolute error). В задачах сегментации изображений по умолчанию в качестве функции ошибки используется коэффициент Дайса[1] (Dice coefficient) или коэффициент Жаккара (Jaccard similarity coefficient). [4, 39, 33]

В рассмотренных ранее моделях ИНС, решающих задачу разделения источников звука, в качестве функции ошибки применяются исключительно сумма квадратов отклонений и сумма модулей отклонений (см. прил. 1). Очевидно, что, опираясь на опыт других исследователей, в рамках настоящей работы должна быть принята одна из перечисленных функций.

Разница между суммой квадратов отклонений и суммой модулей отклонений состоит в том, что первая функция сильнее «наказывает» сеть за бо̀льшѝе ошибки. Таким образом, если появление бо̀льшѝх ошибок является для конкретной задачи критичным, то наиболее предпочтительной является сумма квадратов отклонений. Если в конкретном случае важнее устойчивость модели к выбросам, то предпочтительной является средняя абсолютная ошибка.

В настоящей работе в качестве функции ошибки согласно рекомендациям ряда авторов [4, 2] принята сумма квадратов отклонений.

---

[1] В ряде источников данный коэффициент носит название «коэффициент Сёренсена» (Sorensen–Dice coefficient).

3.3. Алгоритм обучения

Суть обучения нейронных сетей состоит в оптимизации функции ошибки, которая чаще всего является невыпуклой. В настоящее время, алгоритмы обучения за редчайшим исключением в качестве метода оптимизации используют различные модификации градиентного спуска.

Классический градиентный спуск предполагает изменение весов на каждом шаге обучения на основе всего тренировочного множества, что приводит к значительным вычислительным затратам и затрудняет его практическое применение. Стохастический градиентный спуск предусматривает изменение весов не после прохода по всему тренировочному множеству, а после каждого примера. Данный алгоритм обладает рядом достоинств по сравнению с градиентным спуском и может быть применён на практике, однако является не эффективным. Обычно для обучения реальных систем используется стохастический градиентный спуск по мини-батчам[1] (mini-batch), небольшим подмножествам тренировочного набора. Это позволяет пользоваться процедурами матричной арифметики, быстро вычисляемых на видеокарте, сохраняя при этом все достоинства стохастического градиентного спуска. [4]

Важнейшим и самым трудным для установки параметром при обучении ИНС является скорость градиентного спуска, которая регулирует размер шага в направлении склона градиента. При завышенной скорости алгоритм оптимизации никогда не попадёт в минимум, а будет просто прыгать по случайным точкам пространства. В случае слишком маленькой скорости обучение станет очень медленным и алгоритм с большой долей вероятности сойдётся в первом же локальном минимуме.

При применении чистого градиентного спуска скорость обучения и стратегия её изменения задаются вручную. Однако, на сегодняшний день широкое распространение получили адаптивные методы градиентного спуска, учитывающие форму функции ошибки, и варьируемые параметры спуска в зависимости от

[1] В ряде источников используется термин «пакет».

происходящего с функцией. Качественный обзор существующих адаптивных методов приведен в [4, 2] и выходит за рамки настоящей работы. Отметим лишь, что рекомендуемым в указанных источниках методом оптимизации является адаптивный алгоритм оптимизации ADAM, применяемый также в подавляющем большинстве известных решений задачи разделения источников звука (см.прил.1).

Таким образом, с учётом вышесказанного, в качестве метода оптимизации при обучении разрабатываемой ИНС применяется адаптивный вариант стохастического градиентного спуска по мини-батчам – ADAM.

На практике процесс обучения состоит из эпох, состоящих из шагов, каждый из которых состоит из вычисления ошибки на мини-батче и обновления весов сети. Деление на эпохи по сути задаёт шаг, с которым вычисляются метрики, принятые для контроля. В простейшем случае размер эпохи принимается таким, чтобы за одну эпоху обучения был целиком пройден тренировочный набор. Однако при большом объёме набора решение может сойтись до его исчерпания. Таким образом, размер эпохи может быть принят исходя из объективных соображений и предыдущего опыта обучения.

В рамках настоящей работы для обучения разрабатываемой ИНС используется обучающий набор MUSDB18 [42], описанный в главе 4. Тренировочная часть набора содержит ≈5,5 часов размеченных музыкальных записей, что соответствует ≈1,5 миллионам примеров. Исходя из опыта обучения размер эпохи принимается равным 25% тренировочного набора.

Согласно общеизвестным рекомендациям в качестве размера мини-батча следует принимать число равное степени двойки. Оптимальными значениями считаются 32, 64 и 128. В рамках настоящей работы, исходя из опыта, размер мини-батча принят равным 64.

3.4. Начальная инициализация весов

Как было сказано ранее, процесс обучения нейронных сетей состоит в оптимизации функции ошибки, производимой методом градиентного спуска. Начальная инициализация весов или, другими словами, указание точки функции,

из которой следует начинать градиентный спуск, позволяет обучать ИНС быстрее и качественнее.

Доказано [4], что для симметричных функций активации (логический сигмоид, гиперболический тангенс) наилучшим вариантом является инициализацию Ксавье [12], а для ReLU и ему подобных — инициализация Хе [13]. Свободные члены обычно инициализируются нулями.

3.5. Применяемые методы регуляризации

В настоящее время при обучении нейронных сетей широко применяются различные методы модификации алгоритма обучения, называемые регуляризацией и предназначенные для уменьшения ошибки обобщения, без уменьшения ошибки обучения. В рамках настоящего раздела описываются применяемые в рамках настоящей работы методы регуляризации.

3.5.1. Ранняя остановка

Универсальным методом регуляризации является ранняя остановка (early stopping). При её использовании из тренировочного набора выделяется часть (обычно 20%), называемая валидационным набором, которая при обучении используется только для вычисления валидационной ошибки, не влияющей на алгоритм градиентного спуска. Процесс обучения при этом останавливается, когда начинает увеличиваться валидационная ошибка, а не при достижении локального минимума для тренировочного набора.

3.5.2. Дропаут (прореживание)

Дропаут является одним из важнейших методов регуляризации. Его суть заключается в установке для каждого нейрона (кроме нейронов выходного слоя) некоторой вероятности, с которой он может быть исключён из сети. Практика обучения ИНС доказала, что дропаут даёт серьезные улучшения в качестве обученной модели. Это объясняется тем, что дропаут по сути представляет собой своеобразный метод усреднения моделей или беггинга. [4]

3.5.3. Нормализация по мини-батчам (пакетная нормировка)

Нормализация по мини-батчам (batch normalization) является очень мощным методом регуляризации. Решая проблему внутреннего сдвига переменных при обучении (internal covariance shift) она оказывает заметное влияние на качество оптимизации, особенно в случае свёрточных сетей [2].

Суть данного метода заключается в «отбеливании» (whitening) входов каждого слоя сети, то есть приведения их среднего к нулю, а матрицы ковариаций — к единичной. Применение нормализации по мини-батчам позволяет обучать сверхглубокие архитектуры, насчитывающие сотни и тысячи слоёв. [4]

3.6. Архитектура ИНС

Выбор типа архитектуры и её непосредственное проектирование являются ключевыми аспектами при разработке нейронных сетей.

3.6.1. Обоснование типа архитектуры

Выбор типа архитектуры ИНС зависит от решаемой задачи. Если задача заключается в обучении с учителем с векторами фиксированного размера на входе, обычно применяются сети прямого распространения с полносвязными слоями. Если известна топологическая структура входных данных, применяют свёрточные сети. Если входом или выходом являются последовательности, то хорошим выбором может стать рекуррентная сеть. [2]

При решении задачи разделения источников звука, с одной стороны, можно руководствоваться тем, что аудиосигнал, представленный в виде спектрограммы, имеет топологическую структуру, и выбрать свёрточную архитектуру. С другой – аудиосигнал является последовательностью, что является поводом к выбору рекуррентной архитектуры. Как было показано в главе 2, на практике при решении задачи разделения источников звука главным образом применяются свёрточные и рекуррентные ИНС.

Стоит отметить, что на сегодняшний день рекуррентные нейронные сети используются очень широко и являются основной архитектурой для обработки последовательностей. Однако, согласно [4] иногда рекуррентные сети применяются просто по причине их популярности, что приводит к необоснованному

усложнению модели ИНС. Также в [4] отмечается, что в последнее время прослеживается тенденция к замещению рекуррентных сетей свёрточными. Часто оказывается, что свёрточные сети могут решать те же задачи, при том, что обучаются они проще. Заметим также, что согласно [2] на практике успеха обычно добиваются исследователи, правильно применяющие широко известный алгоритм, а не прибегающие к сложному запутанному алгоритму, не до конца понимая его суть.

Опираясь на вышесказанное для реализации в рамках настоящей работы принимается классическая свёрточная архитектура.

3.6.2. Проектирование архитектуры

Большинство нейронных сетей организовано в виде цепочек слоёв, каждый из которых является функцией от предыдущего. Стандартный слой свёрточной сети состоит из трех блоков: свёртка, активация, пулинг (субдискретизация) [4]. Как известно [2], идеальная архитектура сети для конкретной задачи должна определяться в ходе экспериментов, путём контроля ошибки на валидационном наборе[1].

В ходе поиска оптимальной архитектуры ИНС в рамках настоящей работы был проведён ряд экспериментов по обучению различных конфигураций архитектур. В настоящем разделе приводятся результаты обучения наиболее характерных из них.

Обучение каждой из рассматриваемых сетей производилось на протяжении 10 эпох. В качестве функции ошибки в каждом из случаев использовалась сумма квадратов отклонений. В качестве функции активации[2] выходного слоя была принята функция сигмоида, остальных слоёв – функция ReLU.

На первом этапе была произведена попытка обучения двух вариантов классической свёрточной архитектуры (сеть №1 и сеть №2) на мягких частотно-временных масках (см.п.1.3.). Длина отрезов спектрограммы, подающихся на

[1] Процесс подготовки обучающего, валидационного и тестового наборов описан в главе 4.
[2] Обоснование выбора функций активации приведено в п.3.6.3.

вход сетей была принята равной 11 шагам (≈130 мс) дискретного преобразования Фурье по аналогии с [21].

Сеть №1 (см.рис.3.2.) была разработана по аналогии с ИНС [21] с корректировкой её недостатков, а именно заменой каждого из слоёв свёртки с ядром 5×5 на пары последовательных слоёв с ядрами 3×3. Преимущества такой замены были обоснованы ранее в п.2.2.1. Также была произведена корректировка количества ячеек полносвязных слоёв на выходе сети с учётом параметров, принятых в п.3.1.

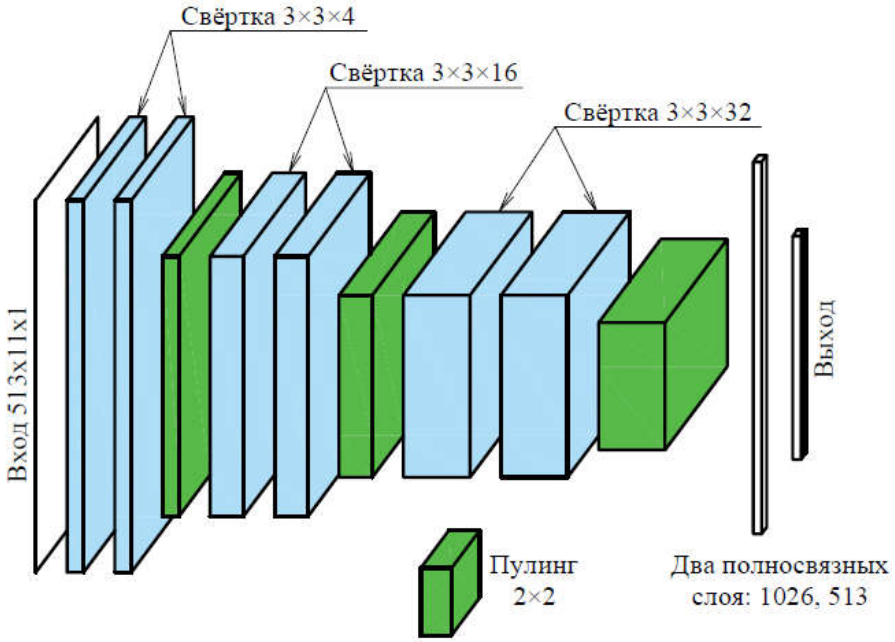

Рис.3.2. Схема архитектуры сети №1.

Эмпирически доказано [2], что для широкого класса задач увеличение глубины ИНС влечет за собой улучшение обобщаемости. Опираясь на этот факт, в качестве сети №2 (см.рис.3.3.) был принят углублённый вариант сети №1, дополненный тремя слоями свёртки с ядром 3×3. Так же по сравнению с сетью №1 было увеличено количество карт признаков по всем свёрточным слоям.

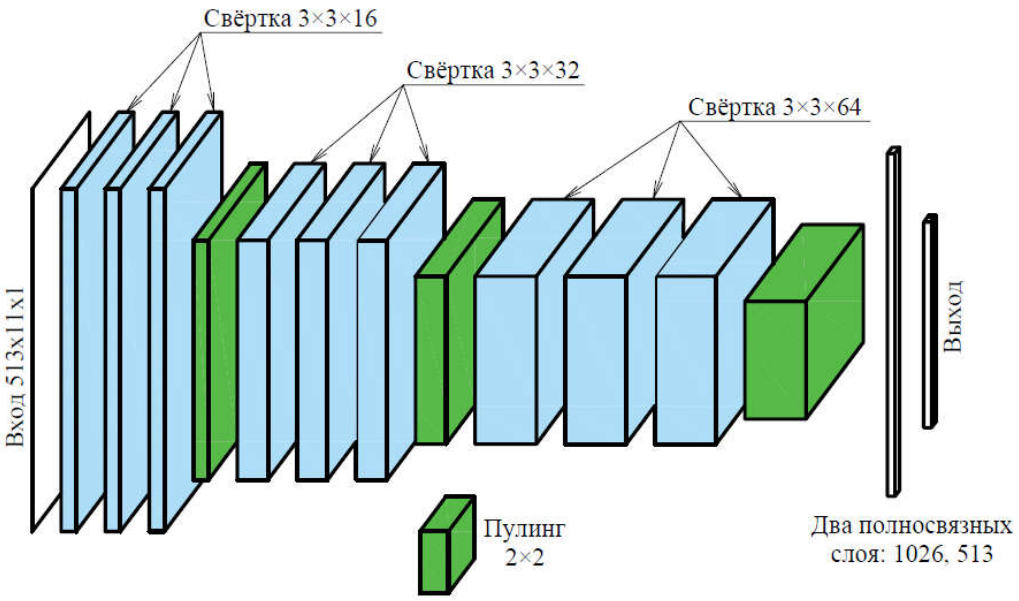

Рис.3.3. Схема архитектуры сети №2.

Графики ошибки на валидационном наборе для описанных архитектур (см.рис.3.4.) имеют скачкообразный характер, по которому нельзя однозначно сделать вывод о снижении соответствующей ошибки в результате обучения. В связи с этим было приято решение о переходе к бинарным частотно-временным маскам с одновременным увеличением длины отрезков спектрограмм, подающихся на вход ИНС.

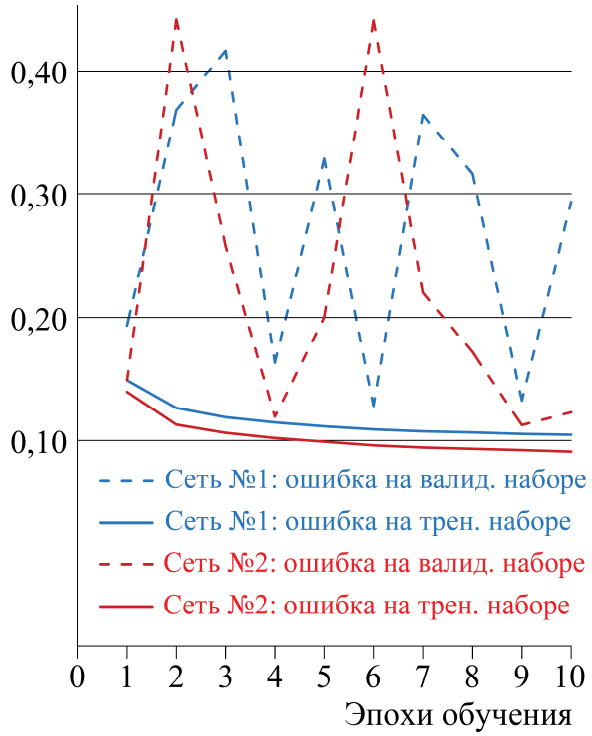

Рис.3.4. Графики обучения сетей №1 и №2.

На втором этапе экспериментального исследования была произведена попытка обучения двух вариантов свёрточной архитектуры (сеть №3 и сеть №4) на бинарных масках частотно-временных масках (см.п.1.3.). Длина отрезов спектрограмм, подающихся на вход сетей была увеличена до 25 шагов (≈290 мс) дискретного преобразования Фурье по аналогии с [36].

Архитектура сети №3 была разработана по типу ИНС описанной в [36] (см.рис.3.5.). Характерной особенностью ИНС [36] является колебание количества карт признаков от слоя к слою. Данное обстоятельство противоречит установившимся правилам создания архитектур глубоких свёрточных сетей, согласно которым число карт признаков должно постепенно расти на более глубоких уровнях [4].

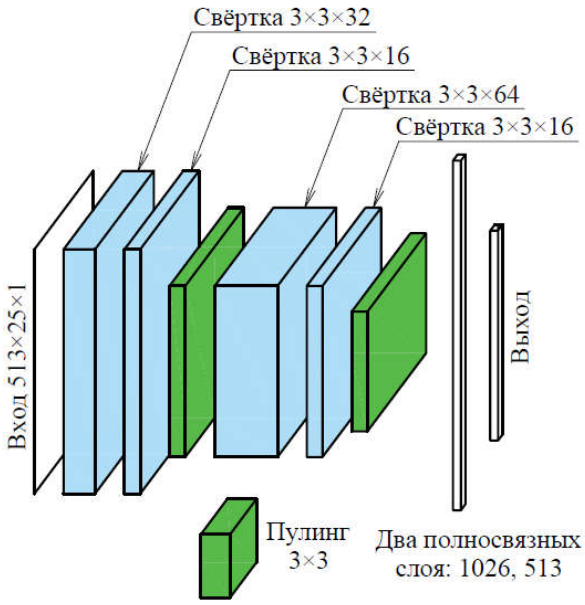

Рис.3.5. Схема архитектуры сети №3.

Для оценки влияния колебания количества карт признаков последовательных свёрточных слоёв на качество обучения была разработана сеть №4 (см.рис.3.6.), число карт признаков слоёв свёртки которой растёт с глубиной при прочих характеристиках аналогичных сети №3.

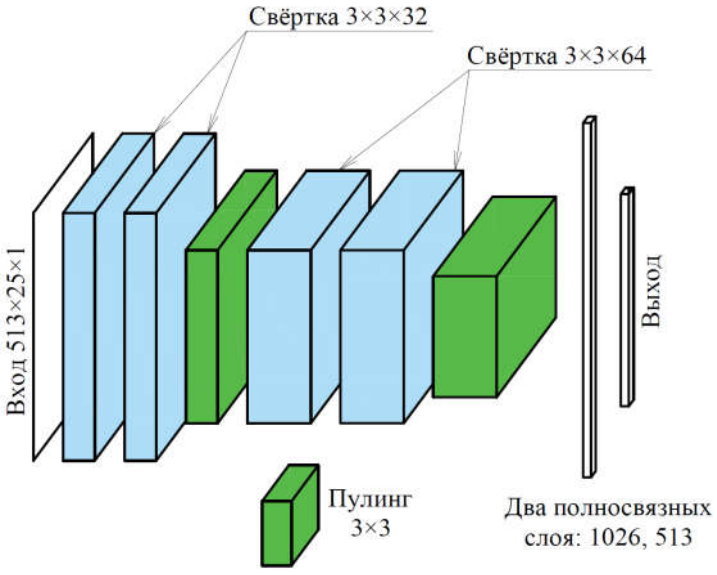

Рис.3.6. Схема архитектуры сети №4.

Форма графиков ошибки на валидационном наборе для сетей №3 и №4 (см.рис.3.7.) позволяют сделать заключение о том, что соответствующая ошибка снижается в ходе обучения. Иными словами, можно утверждать, что в отличии от рассмотренных ранее сетей №1 и №2 сети №3 и №4 действительно обучаются. Очевидно, этому способствует использование бинарных масок, облегчающее задачу ИНС, а также увеличение длины отрезков спектрограмм на входе, обеспечивающее более широкий контекст до и после целевого единичного отрезка (см.рис.3.1.). Так же, важно отметить, что колебание числа карт признаков на последовательных слоях свёртки, принятое в архитектуре сети №3 по аналогии с [36], не оказывает положительного эффекта на величину ошибки на тренировочном и тестовом наборах.

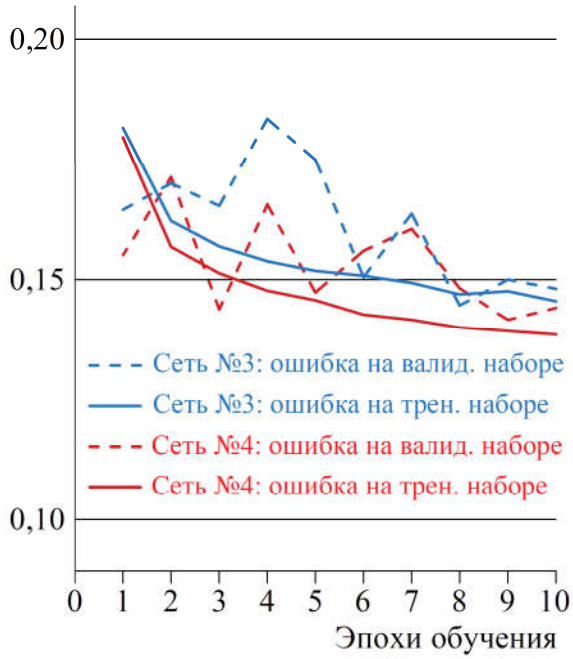

Рис.3.7. Графики обучения сетей №3 и №4.

С учётом выводов, полученных в результате описанных экспериментов, для реализации в качестве основного решения была принята архитектура, представленная на рисунке 3.8. Три пары последовательных свёрточных слоёв сети с ядрами 2×2 завершаются соответственно слоями пулинга с ядрами 3×3. Для возможности применения трёх слоёв пулинга с указанным размером ядра на вход

сети подаётся отрезок спектрограммы длиной 27 шагов (≈310 мс) оконного преобразования Фурье. На выходе сети предусмотрены два полносвязных слоя, содержащих 1026 и 513 нейронов. В качестве функции активации выходного слоя принята функция сигмоида, остальных слоёв – функция ReLU[1]. Архитектура разработанной ИНС в виде текстового описания, сформированного посредством функций библиотеки Keras, представлена в приложении 2.

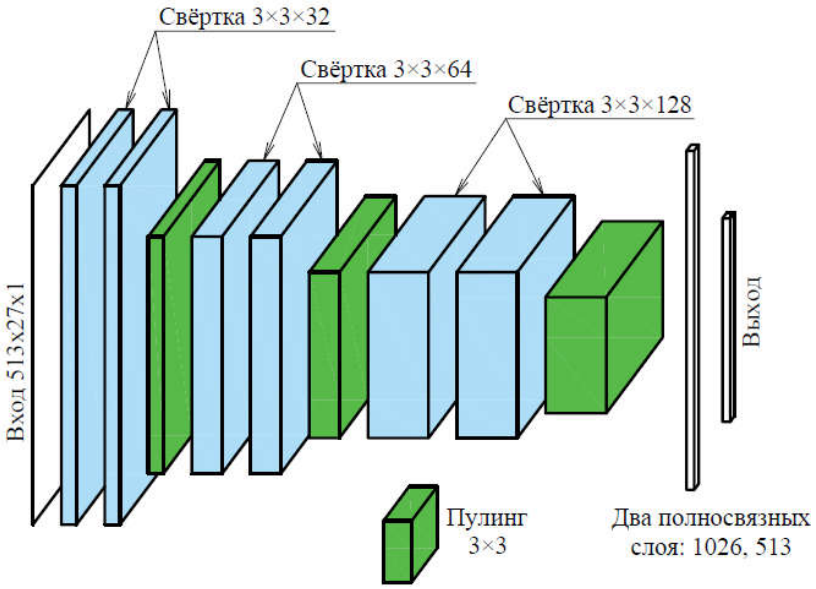

Рис.3.8. Схема итоговой архитектуры ИНС.

На рисунке 3.9. представлены графики функции ошибки на обучающем и валидационном наборах для итоговой сети, а также для сетей №3 и №4. По графикам видно, что ошибка на валидационном наборе для итоговой сети снижается быстрее, чем у сетей №3 и №4. Иными словами, итоговая сеть обучается лучше остальных.

[1] Обоснование выбора функций активации приведено в п.3.6.3.

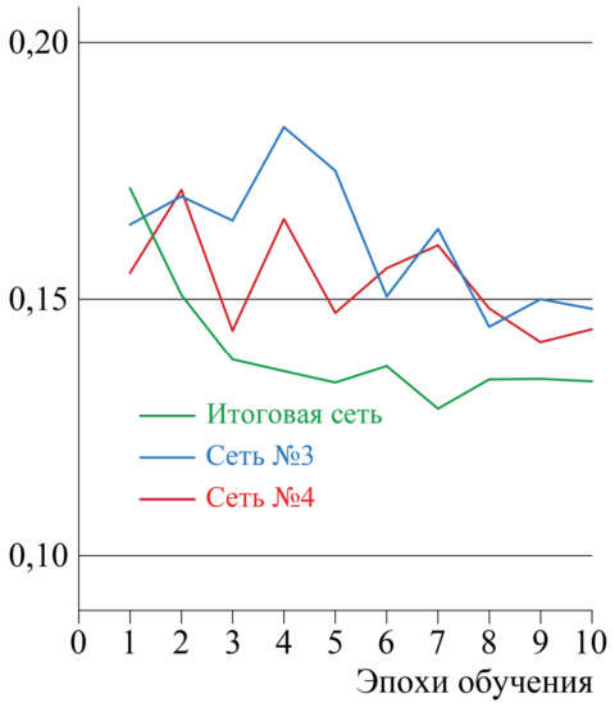

Рис.3.9. Графики функции ошибки на валидационном наборе итоговой сети, а также сетей №3 и №4.

3.6.3. Функции активации

На сегодняшний день существует множество различных функций активации. Большинство источников рекомендуют по умолчанию принимать в качестве функции активации рядовых слоёв ReLU (rectified linear unit), описываемую уравнением $y(x) = max\{0, x\}$ или её общения, например, LeakyReLU (leaky rectified linear unit), описываемую уравнением $y(x) = max\{k \cdot x, x\}$. Доказано, что нейроны, использующие данные функции активации, эффективнее основанных на логистическом сигмоиде и гиперболическом тангенсе. [4, 2] Опираясь на вышесказанное, в качестве функции активации рядовых слоёв сети принята функция ReLU.

В качестве функции активации выходного слоя разрабатываемой сети принята функция сигмоида, описываемая уравнением $y(x) = \frac{1}{1+e^{-x}}$. Данное решение обосновывается тем, что область значений данной функции составляет

(0;1). При этом, значения элементов частотно-временных масок, отрезки которых служат ответами в обучающих примерах, находятся в диапазоне [0;1]. Таким образом, использование на выходном слое функции сигмоида позволяет облегчить задачу нейронной сети при обучении.

С.И. Николенко [4] отмечает, что выбор функции активации не является главным вопросом в разработке ИНС; архитектура системы и алгоритмы оптимизации играют гораздо более важную роль.

ГЛАВА 4. ПОДГОТОВКА ОБУЧАЮЩЕГО, ВАЛИДАЦИОННОГО И ТЕСТОВОГО НАБОРОВ

Качественный и достаточный по объёму тренировочный набор имеет решающее значение при обучении ИНС. Для задач классификации существует грубое эвристическое правило, согласно которому алгоритм обучения достигает приемлемого качества при наличии примерно 5000 размеченных примеров на категорию и может быть сопоставим или превзойти человека при наличии не менее 10 миллионов примеров [2]. Решаемая задача относится к регрессии, тем не менее порядок приведённых чисел может служить ориентиром.

4.1. Описание обучающего набора MUSDB18

Разрабатываемая в рамках настоящей работы ИНС обучается на наборе MUSDB18 [42], который является официальным набором Международной кампании по оценке качества разделения источников (Signal Separation Evaluation Campaign (SiSEC)) [44]. Набор используется с разрешения авторов: З. Рафии, А. Лиуткуса, Ф.Р. Штетера, С.И. Мимилакиса, Р. Биттнера.

Описываемый обучающий набор состоит из 150 музыкальных композиций общей длительностью порядка 10 часов. Все композиции представлены в многодорожечном формате «mp4» и состоят из пяти стереофонических дорожек, соответствующих смешанному сигналу, перкуссии, басу, вокалу и остальному. Поскольку целью данной работы является разработка алгоритма, работающего с монофоническим аудиосигналом, в процессе извлечения дорожек производится их конвертация из стерео в моно. Так же стоит напомнить, что согласно сказанному в п.3.1, с целью уменьшения объёма вычислений при подготовке данных производится субдискретизация с 44100 (Гц) до 22050 (Гц).

Набор MUSDB18 разбит авторами на тренировочное, валидационное и тестовое подмножества. Тестовый набор составляет 40% от общего числа записей, остальные распределены между тренировочным и валидационным наборами в процентном соотношении 80/20. Таким образом, с учётом принятых в п.3.1 характеристик тренировочное множество состоит из ≈1,5 миллионов примеров.

4.2. Выбор целевого источника

Как показывает практика при решении задач разделения источников звука посредством ИНС для изоляции каждого из различных источников обучается свой экземпляр сети. В рамках настоящей работы в качестве целевого источника принят вокал. Это объясняется следующими соображениями. Во-первых, согласно [20] вокал и перкуссия являются наиболее часто встречающимися инструментами в западной популярной музыке. Во-вторых, вокал и перкуссия являются составными источниками[1], что несомненно усложняет задачу их выделения. В-третьих, согласно отчёту [44] качество существующих на сегодняшний день решений задачи выделения вокала является недостаточно удовлетворительным и требует улучшения.

[1] Под составным источником звука понимается несколько физических источников (барабан, тарелка, бубен или глухие согласные и гласные звуки) объединённых номинально в один.

ГЛАВА 5. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ СЕТИ

5.1. Перечень и обоснование использованных инструментов

В качестве языка реализации в рамках настоящей работы был принят язык Python, являющийся на сегодняшний день бесспорным лидером в сфере машинного обучения (см.рис.5.1) и одним из наиболее популярных языков программирования в целом [45]. Помимо прочих факторов популярность Python в сфере машинного обучения обусловлена наличием в его экосистеме двух наиболее популярных в настоящее время библиотек автоматического дифференцирования TensorFlow и Theano а также библиотеки Keras, являющейся надстройкой над ними и предназначенной непосредственно для работы с нейронными сетями.

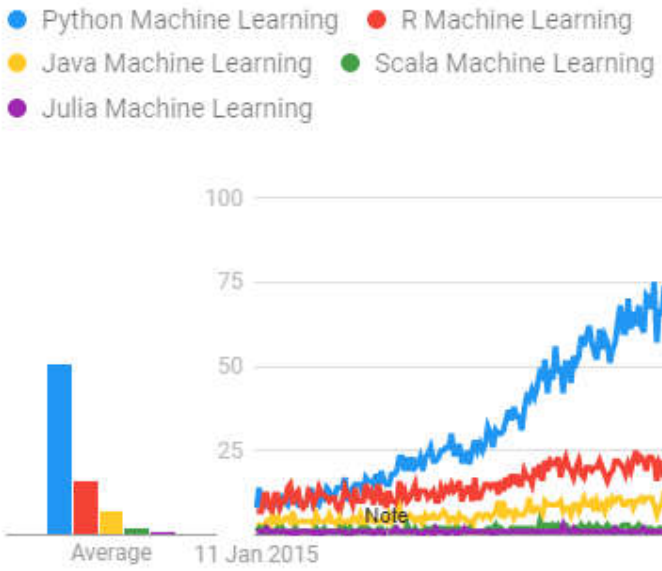

Рис.5.1. Количество поисковых запросов связанных с языками программирования, применяющимися в машинном обучении согласно Google Trends [38].

В рамках настоящей работы в процессе создания модели ИНС и её обучения была применена библиотека Keras с TensorFlow в качестве бэкенда. Для работы с аудиосигналами, в частности для осуществления прямого и обратного преобразований Фурье, была использована специализированная библиотека LibROSA [40]. Для оценки качества разделения источников звука применена

библиотека mir_eval. Для удобства парсинга обучающего набора применён разработанный его авторами пакет musdb.

Стоит отметить, что при одиночной установке пакетов различных Python библиотек неизбежно возникают зависимости и конфликты, разрешение которых является нетривиальной задачей. Во избежание описанных трудностей был использован популярный дистрибутив Anaconda [35], осуществляющий поставку наиболее востребованных библиотек единым согласованным комплектом и содержащий менеджер пакетов Conda [37], предназначенный для установки пакетов с автоматической обработкой зависимостей.

В качестве среды разработки применялась IDE Spyder, входящая в состав дистрибутива Anaconda.

5.2. Детали реализации

Программный код, написанный на языке Python в рамках настоящей работы, для удобства разбит на несколько модулей:

- bss_cnn.py, содержащий единственную функцию, внутри которой происходит инициализация и компиляция модели ИНС;
- generator.py, содержащий описание класса-генератора, формирующего батчи (пакеты примеров) из аудиофайлов обучающего набора, находящихся на жёстком диске;
- train.py, являющийся основным модулем, внутри которого осуществляется задание глобальных параметров, вызов функции создания сети из cnn.py, запуск процесса обучения, вывод графиков обучения и сохранение итоговой модели ИНС;
- predict.py, содержащий код для получения аудиосигнала соответствующего искомому источнику из смешанного аудиосигнала на основе предсказаний обученной сети;
- eval.py, содержащий код оценки качества разделения источников звука.

Листинги исходного кода описанных модулей приведены в приложении 2.

5.3. Оценка результатов обучения

Результаты обучения разработанной сети прежде всего можно оценить по графикам функции ошибки на тренировочном и валидационном наборах представленном на рисунке 5.2. Форма обоих графиков свидетельствует о постепенном снижении соответствующих ошибок. Иными словами, ИНС успешно обучается.

Также можно сделать вывод о том, что данная сеть может быть дообучена, так как не наблюдается прекращения уменьшения ошибки на тренировочном наборе или существенного роста ошибки на валидационном наборе. Разработанная ИНС не была дообучена в рамках работы по причине ограничений в вычислительных и временных ресурсах.

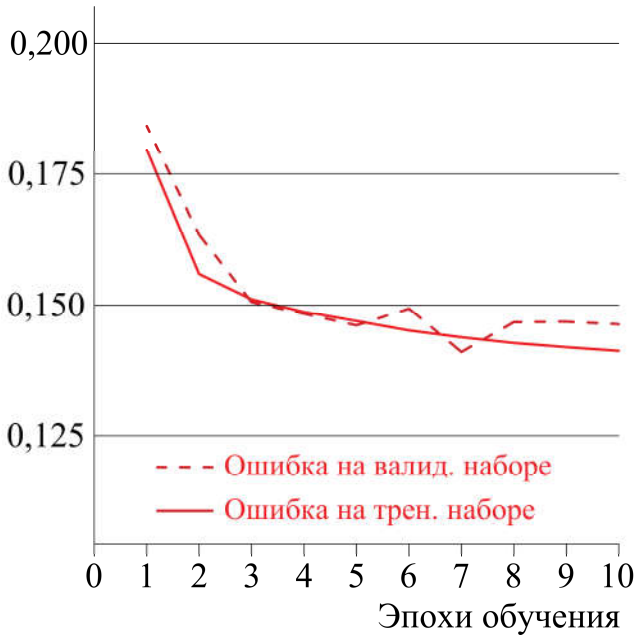

Рис.5.2. График функции ошибки на тренировочном и валидационном наборах для обученной ИНС.

Помимо оценки графиков функций ошибки результат обучения ИНС безусловно должен контролироваться путём «визуализации» модели в действии. То есть, для алгоритма разделения источников звука следует буквально прослушивать сигнал искомого источника, полученный с помощью обученной сети. С

одной стороны, безусловно, данная оценка носит сугубо субъективный характер, с другой – согласно мнению ряда авторитетных исследователей [2] прямое наблюдение за тем, как ИНС справляется со поставленной перед ней задачей, помогает установить, являются ли достигнутые количественные показатели разумными. Таким образом, основываясь на вышесказанном, один из этапов оценки результатов обучения разработанной ИНС заключался в выделении сигнала искомого источника для нескольких музыкальных композиций и их прослушивании.

Заключительным и самым важным этапом оценки результатов обучения ИНС является оценка эффективности всего алгоритма по соответствующим общепринятым метрикам. Данный вопрос освящён в следующей главе настоящей работы.

ГЛАВА 6. ОЦЕНКА ЭФФЕКТИВНОСТИ ПО ОБЪЕКТИВНЫМ КРИТЕРИЯМ

6.1. Применяемые метрики

Очевидно, что оценка эффективности принятых в исследовательских задачах решений должна проводиться по общепринятым в соответствующих областях метрикам. При этом сравнение различных алгоритмов желательно производить с использованием одного и того же набора данных.

В 1999 году Д. Шоббеном и К. Торккола [23] впервые был предложен набор стандартных метрик для оценки алгоритмов слепого разделения источников. До этого оценки часто основывались на косвенных измерениях производительности или субъективных тестах. При этом зачастую разные алгоритмы использовали разные оценки, что безусловно затрудняло возможность их сравнения. [20]

Если говорить непосредственно об области слепого разделения источников звука, то общепринятым в настоящее время считается набор метрик, предложенных Э. Винсентом, Р. Грибонвалем, С. Февоттом [29]. В основе предложенных метрик лежит идея о том, что общая ошибка e_{total} между реальным сигналом выделяемого источника s и сигналом, полученным в результате работы алгоритма $\hat{s}$ состоит из трёх составляющих: ошибки от проникновения в итоговый сигнал компонентов других источников e_{interf}; ошибки от шума e_{noise}; артефактов, появившихся по вине алгоритма e_{artif}. В виде формулы: $\hat{s} = s_{target} + + e_{total} = s_{target} + e_{interf} + e_{noise} + e_{artif}$.

Соответственно, в настоящее время используются следующие метрики:

— Source to Distortion Ratio: $\mathrm{SDR} = \frac{\|s_{target}\|^2}{\|e_{interf}+e_{noise}+e_{artif}\|^2}$;

— Source to Interferences Ratio: $\mathrm{SIR} = \frac{\|s_{target}\|^2}{\|e_{interf}\|^2}$;

— Sources to Noise Ratio: $\mathrm{SNR} = \frac{\|s_{target}+e_{interf}\|^2}{\|e_{noise}\|^2}$;

— Sources to Artifacts Ratio: $\mathrm{SAR} = \frac{\|s_{target}+e_{interf}+e_{noise}\|^2}{\|e_{artif}\|^2}$.

Наиболее часто применяемой метрикой является коэффициент отношения сигнала к искажениям (Source to Distortion Ratio (SDR)), который представляет общую оценку качества разделения. Таким образом, в рамках настоящей работы сравнение разработанного алгоритма с известными решениями производится на основе коэффициента отношения сигнала к искажениям.

Сравнение разработанного алгоритма производится с решениями участников Международной кампании по оценке качества разделения источников (Signal Separation Evaluation Campaign (SiSEC)) [44].

6.2. Способы обработки выходов ИНС

Существует несколько способов обработки выходов обученной нейронной сети. При этом очевидно, что выбор конкретного способа обработки напрямую влияет на значения оценочные метрики.

Если говорить конкретно о решаемой задаче, то самым простым способом обработки выходов обученной ИНС является использование выходов сети как мягкой маски.

Более логичным при обучении ИНС на бинарных масках способом является приведение выходного вектора соответственно к бинарной маске. Для этого каждый элемент выходного вектора сравнивается с некоторым коэффициентом, лежащем в диапазоне (0;1), и по умолчанию равным 0,5. Если рассматриваемый элемент больше данного коэффициента, то он заменяется на «1», если меньше – на «0».

Ещё одним оригинальным способом обработки выходов сети является обнуление в выходном векторе элементов, значение которых меньше некоторого коэффициента, лежащего в диапазоне (0;1) и по умолчанию равного 0,5. Таким образом, из выходных векторов сети будет получена мягкая маска с обнулением части значений.

Пример обработки выходов ИНС различными способами представлен на рисунке 6.1.

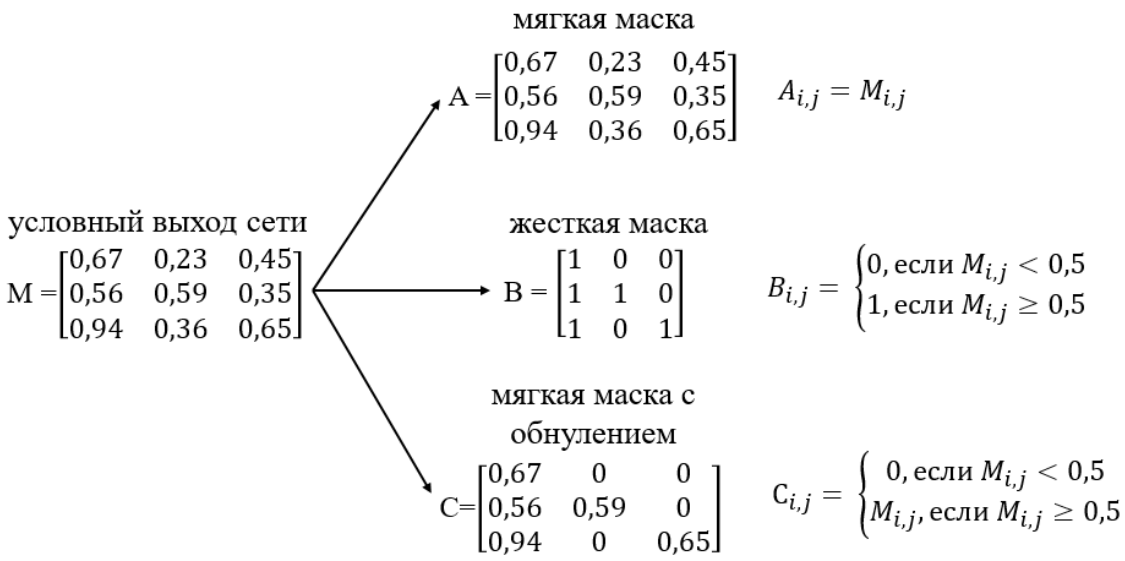

Рис.6.1. Способы обработки выходов ИНС.

В рамках настоящей работы при оценке качества разделения источников звука производится приведение выходов сети к бинарной маске.

6.3. Сравнение разработанного алгоритма с известными решениями

Результат сравнения разработанного алгоритма с решениями участников Международной кампании по оценке качества разделения источников (Signal Separation Evaluation Campaign (SiSEC)) [44] представлен на рисунке 6.2. Диаграмма размаха SDR для разработанного алгоритма обозначена красным цветом. Слева на рисунке указаны названия алгоритмов. Информацию об авторах и ссылки на соответствующие источники можно найти в [44].

Согласно результатам сравнения, разработанный алгоритм разделения источников звука, по качеству превосходит все алгоритмы, основанные не на ИНС, а также ряд алгоритмов на основе ИНС.

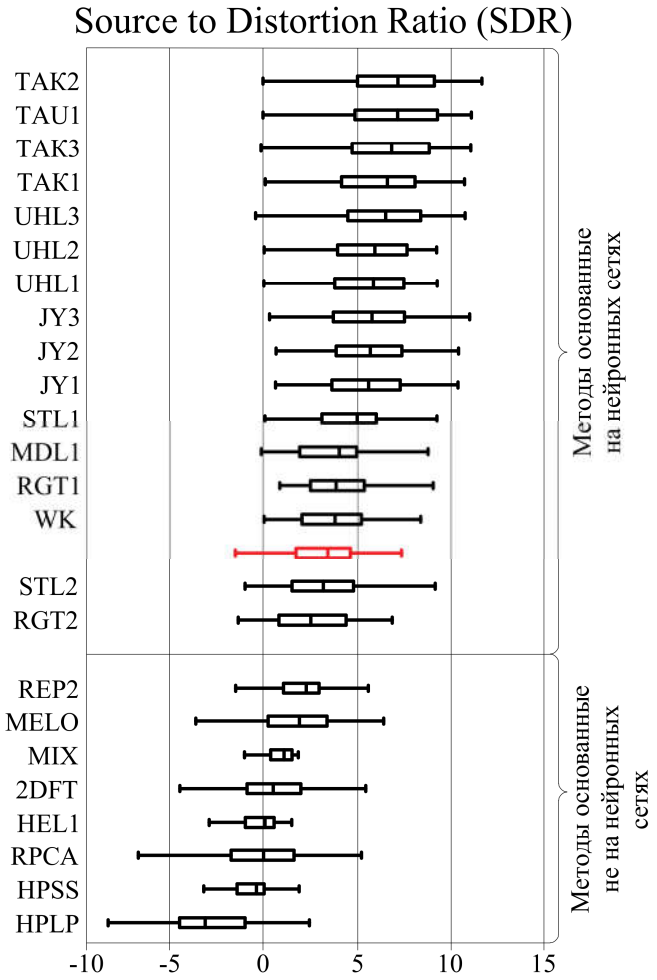

Рис.6.2. Сравнение качества разработанного алгоритма с решениями участников Международной кампании по оценке качества разделения источников [44].

ГЛАВА 7. ЗАКЛЮЧЕНИЕ

В ходе настоящей работы цель, поставленная во введении, была достигнута. А именно, был разработан алгоритм слепого разделения источников звука в монофоническом музыкальном аудиосигнале с использованием искусственных нейронных сетей.

При этом были решены следующие задачи:

- Произведён обзор методов разделения источников звука, в частности в музыкальном аудиосигнале;
- Произведён обзор архитектур ИНС, применяющихся в задачах разделения источников звука;
- Разработана архитектура ИНС, произведено описание алгоритма обучения;
- Осуществлена подготовка обучающего, валидационного и тестового наборов;
- Осуществлена реализация разработанной архитектуры;
- Произведена оценка эффективности по объективным критериям.

В ходе проведенных исследований был сделан ряд следующих выводов. Обучение ИНС на бинарных частотно-временных масках, по сравнению с мягкими масками, существенно облегчает задачу сети. Увеличение длины отрезков спектрограмм, подающихся на вход сети, оказывает положительное влияние на качество решения. Увеличение числа карт признаков на более глубоких уровнях ИНС является оптимальным для ИНС, решающих задачу разделения источников звука.

В заключение стоит отметить, что существует ряд способов повышения эффективности разработанного алгоритма. Во-первых, обучаемая на первом этапе на бинарных масках сеть может быть дообучена на мягких масках. Во-вторых, могут быть применены различные способы обработки выходов ИНС, описанные в п.6.2.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Горячкин, О.В. Методы слепой обработки сигналов и их приложения в системах радиотехники и связи [Текст]. — М.: Радио и связь, 2003. – 230 с.
2. Гудфеллоу, Я. Глубокое обучение / Я.Гудфеллоу, И.Бенджио, А.Курвилль / пер. с анг. А.А.Слинкина. – 2-е изд., испр. – М.: ДМК Пресс, 2018. – 652 с.
3. Ковалёва, А.А. Разработка эффективного метода идентификации человека по голосу. [Текст]: ВКР магистра: 02.04.03 / Ковалёва Александра Алексеевна. – СПб., 2018. – 89 с.
4. Николенко, С. Глубокое обучение [Текст] / С.Николенко, А.Кадурин, Е.Архангельская — СПб.: Питер, 2018. – 480 с.
5. Шульгин В. И., Морозов А. В., Волосюк Е. В. Использование технологии «слепого разделения источников» при обработке биомедицинских сигналов // Клиническая информатика и Телемедицина 1/2005. – Харьков, Украина, 2005 – С.42-50.
6. Bengio Y. Learning Deep Architectures for AI / / Foundations and Trends in Machine Learning, 2009, vol. 2, no. 1. — P. 1-127.
7. Felipe do Carmo , Joaquim T. de Assis, Vania Vieira Estrela, Alessandra M.Coelho. Blind signal separation and identification of mixtures of images // Universidade do Estado do Rio de Janeiro (UERJ), Instituto Politécnico do Rio de Janeiro (IPRJ), Nova Friburgo, RJ, Brazil.
8. Cherry, C. Some Experiments on the Recognition of Speech, with One and with Two Ears. The Journal of the Acoustical Society of America, 1953.
9. Doire C. S. J. Online singing voice separation using a recurrent one-dimensional u-net trained with deep feature losses.
10. FitzGerald, D. The Good Vibrations Problem, AES 134th Convention, Rome, Italy, 2013 May 4–7.
11. Fitzgerald, D., Cranitch, M. & Coyle, E. (2005) Non-negative Tensor Factorisation for Sound Source Separation, Proceedings of the Irish Signals and Systems Conference, Dublin, Ireland, 2005.

12. Glorot X., Bengio Y. Understanding the Difficulty of Training Deep Feedforward Neural Networks / / International conference on artificial intelligence and statistics, 2010.
13. He K. et al. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification / / / Proc. ICCV 2015, 2015.
14. Heinzel G., Rudiger A., Schilling R. Spectrum and spectral density estimation by the Discrete Fourier transform (DFT), including a comprehensive list of window functions and some new flat-top windows. Max-Planck-Institut fur ¨ Gravitationsphysik.
15. Hinton G. E. Learning Multiple Layers of Representation / / Trends in Cognitive Sciences, 2007, vol. 11.-P . 428-434.
16. Hinton G. E., Osindero S., Teh Y.-W. A Fast Learning Algorithm for Deep Belief Nets / / Neural Computation, 2006, vol. 18, no. 7.
17. Jansson, A., Humphrey, E., Montecchio, N., Bittner, R., Kumar, A. and Weyde, T. ORCID: 0000-0001-8028-9905 (2017). Singing voice separation with deep U-Net convolutional networks. Paper presented at the 18th International Society for Music Information Retrieval Conference, 23-27 Oct 2017, Suzhou, China
18. Mimilakis S.I., Drossosy K., Santos J.F., Schuller G., Virtaneny T., Bengio Y.. Monaural Singing Voice Separation with Skip-Filtering Connections and Recurrent Inference of Time-Frequency Mask.
19. Noh H., Hong S., Han B. Learning deconvolution network for semantic segmentation. In Proceedings of the IEEE International Conference on Computer Vision, pages 1520–1528, 2015.
20. Piñon, R.M. Audio Source Separation for Music in Low-latency and High-latency Scenarios: tesi doctoral upf. – Universitat Pompeu Fabra, 2013.
21. Roma G., Green O., Tremblay P.A., Improving single-network single-channel separation of musical audio with convolutional layers. Proceedings of LVA/ICA, 2018
22. Ronneberger O., Fischer P., Brox T. "U-Net: Convolutional Networks for Biomedical Image Segmentation", 2015

23. Schobben D. and Torkkola K. Evaluation of blind signal separation metods. Proc Int Workshop on ICA and, 5, 1999.
24. Srivastava R.K., Greff K., Schmidhuber/. T raining Very Deep Networks / / Proc. 28th NIPS, Cambridge, MA, USA: MIT Press, 2015. - P. 2377-2385.
25. Stoller D., Ewert S., Dixon S.. Wave-U-Net: A Multi-Scale Neural Network for End-to-End Audio Source Separation.
26. Takahashi N., Mitsufuji Y. Multi-scale multi-band DenseNets for audio source separation, Proc. WASPAA, 2017
27. Uhlich S., Porcu M., Giron F., Enenkl M., Kemp T., Takahashi N., Mitsufuji Y. “Improving music source separation based on deep neural networks through data augmentation and network blending”, Proc. ICASSP, 2017.
28. Vincent E., Gribonval R., Plumbley M. Oracle estimators for the benchmarking of source separation algorithms. Signal Processing, Elsevier, 2007.
29. Vincent E., Gribonval R., Fevotte C., “Performance measurement in blind audio source separation,” IEEE Trans. Speech and Audio Proc., 2005, to appear.
30. Vincent E., Fevotte C. , Gribonval R., Benaroya L., Rodet X., Röbel A., Carpentier E.L., and Bimbot F. A tentative typology of audio source separation tasks. In 4th Int. Symp. on Independent Component Analysis and Blind Signal Separation (ICA), pages 715-720, Nara, Japan, 2003.
31. Ward D., Takahashi Q.K.N., Goswami N., Mitsufuji Y. MMDeseLSTM: an efficient combination of convolutional and recurrent neural networks for audio source separation
32. Простыми словами о преобразовании Фурье: [Электронный документ]. – (https://habr.com/ru/post/196374/). Проверено 11.01.2020.
33. Сегментация изображений при помощи нейронной сети U-Net: [Электронный документ]. – (http://robocraft.ru/blog/machinelearning/3671.html). Проверено 11.01.2020.
34. Сибилянты: [Электронный документ]. – (https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%B1%D0%B8%D0%BB%D1%8F%D0%BD%D1%82%D1%8B). Проверено 11.01.2020.

35. Anacaonda | The World’s Most Popular Data Science Platform: [Электронный документ]. – (https://www.anaconda.com/). Проверено 11.01.2020.
36. Audio AI: isolating vocals from stereo music using Convolutional Neural Networks: [Электронный документ]. – (https://towardsdatascience.com/ audio-ai-isolating-vocals-from-stereo-music-using-convolutional-neural-networks-210532383785) Проверено 21.10.2019.
37. Conda – Conda documentation: [Электронный документ]. – (https://docs.conda.io/en/latest/). Проверено 11.01.2020.
38. Google Trends: [Электронный документ]. – (https://trends.google.com/trends/explore?date=today%205-y&q=Python%20Machine%20Learning,R%20Machine%20Learning,Java%20Machine%20Learning,Scala%20Machine%20Learning,Julia%20Machine%20Learning). Проверено 11.01.2020.
39. How to Choose Loss Functions When Training Deep Learning Neural Networks: [Электронный документ]. – (https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/). Проверено 11.01.2020.
40. LibROSA | librosa 0.7.1 documentation: [Электронный документ]. – (https://librosa.github.io/librosa/). Проверено 11.01.2020.
41. LOUD: Large acOUstic Data Array Project: [Электронный документ]. – (http://groups.csail.mit.edu/cag/mic-array/) Проверено 25.05.2019.
42. MUSDB18 | SigSep: [Электронный документ]. – (https://sigsep.github.io/datasets/musdb.html). Проверено 11.01.2020.
43. Single-Channel Source Separation Tutorial Mini-Series by Nicholas Bryan, Dennis Sun, and Eunjoon Cho: [Электронный документ]. – (https://ccrma.stanford.edu/~njb/teaching/sstutorial) Проверено 03.10.2019.
44. The 2018 Signal Separation Evaluation Campaign: [Электронный документ]. – (https://arxiv.org/abs/1804.06267) Проверено 21.10.2019.
45. TIOBE Index for January 2020: [Электронный документ]. – (https://www.tiobe.com/tiobe-index/). Проверено 11.01.2020.

46. Wang, D. Deep Learning Reinvents the Hearing Aid: [Электронный документ]. – (https://spectrum.ieee.org/consumer-electronics/audiovideo/deep-learning-reinvents-the-hearing-aid) Проверено 02.10.2019.

ПРИЛОЖЕНИЕ 1.

Сравнение известных решений задачи разделения

источников в музыкальном аудиосигнале

Авторы	Архитектура	Функция ошибки	Оптимизатор	Частота дискретизации	Размер окна преобразования Фурье	Размер шага преобразования Фурье	Вид маски
Ж. Рома, О. Грин, П.А.Трамбле [21]	Классическая светрочная	сумма квадратов отклонений	ADAM	22,05	256	2048	мягкая
Э. Корецкий [36]	Классическая светрочная	сумма квадратов отклонений	SGD	22,05	256	1024	бинарная
А. Янссон, Э. Хамфри, Н. Монтеккио, Р. Биттнер, А. Кумар, Т. Вейд [17]	Свёрточная U-Net архитектура	средняя абсолютная ошибка	ADAM	8,19	768	1024	мягкая
Д. Столлер, С. Эверт, С. Диксон [25]	Wave-U-Net, одномерная адаптация U-Net	сумма квадратов отклонений	ADAM	22,05	-	-	-
С. Улич, М. Порку, Ф. Хирон, М. Эненкл, Т. Кемп, Ю. Мицуфуджи, Н. Такахаши [27]	Сеть с долгой краткосрочной памятью	средняя абсолютная ошибка	ADAM	44,1	1024	4096	мягкая
Д. Уорд, Ц. Ц. Конг [31]	Сеть с долгой краткосрочной памятью	средняя абсолютная ошибка	ADAM	32	512	1024	-
С.И. Мимилакис, К. Дроссоси, Ж.Ф. Сантос, Д. Шуллер, Т. Виртанени, И. Бенджио[18]	Рекуррентная архитектура кодировщик-декодировщик	сумма квадратов отклонений	ADAM	44,1	384	4096	мягкая
Н. Такахаси, Ю. Мицуфуджи, Н. Госвами [26]	Рекуррентная U-Net Архитектура	сумма квадратов отклонений	ADAM	11	1024	4096	бинарная
К. Дур [9]	Рекуррентная U-Net Архитектура	средняя абсолютная ошибка	RMSprop	44,10	512	2048	мягкая

ПРИЛОЖЕНИЕ 2.

Архитектура ИНС в виде текстового описания,
сформированного посредством библиотеки Keras

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 513, 27, 32)	320
activation_1 (Activation)	(None, 513, 27, 32)	0
batch_normalization_1	(Batch (None, 513, 27, 32)	128
conv2d_2 (Conv2D)	(None, 513, 27, 32)	9248
activation_2 (Activation)	(None, 513, 27, 32)	0
batch_normalization_2	(Batch (None, 513, 27, 32)	128
max_pooling2d_1	(MaxPooling2 (None, 171, 9, 32)	0
dropout_1 (Dropout)	(None, 171, 9, 32)	0
conv2d_3 (Conv2D)	(None, 171, 9, 64)	18496
activation_3 (Activation)	(None, 171, 9, 64)	0
batch_normalization_3	(Batch (None, 171, 9, 64)	256
conv2d_4 (Conv2D)	(None, 171, 9, 64)	36928
activation_4 (Activation)	(None, 171, 9, 64)	0
batch_normalization_4	(Batch (None, 171, 9, 64)	256
max_pooling2d_2	(MaxPooling2 (None, 57, 3, 64)	0
dropout_2 (Dropout)	(None, 57, 3, 64)	0
conv2d_5 (Conv2D)	(None, 57, 3, 128)	73856
activation_5 (Activation)	(None, 57, 3, 128)	0

batch_normalization_5	(Batch (None, 57, 3, 128)	512
conv2d_6 (Conv2D)	(None, 57, 3, 128)	147584
activation_6 (Activation)	(None, 57, 3, 128)	0
batch_normalization_6	(Batch (None, 57, 3, 128)	512
max_pooling2d_3	(MaxPooling2 (None, 19, 1, 128)	0
dropout_3 (Dropout)	(None, 19, 1, 128)	0
flatten_1 (Flatten)	(None, 2432)	0
dense_1 (Dense)	(None, 1026)	2496258
activation_7 (Activation)	(None, 1026)	0
batch_normalization_7	(Batch (None, 1026)	4104
dropout_4 (Dropout)	(None, 1026)	0
dense_2 (Dense)	(None, 513)	526851
activation_8 (Activation)	(None, 513)	0

Total params: 3,315,437
Trainable params: 3,312,489
Non-trainable params: 2,948

ПРИЛОЖЕНИЕ 3.

Листинги исходного кода

Пакет bss_cnn.py

```
from keras.models import Sequential
from keras.layers.normalization import BatchNormalization
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.layers.core import Activation
from keras.layers.core import Flatten
from keras.layers.core import Dropout
from keras.layers.core import Dense
from keras import backend as K
import keras.initializers

class BSS_CNN:
    @staticmethod
    def define(freq_bins, length):
        model = Sequential()

        inputShape = (freq_bins, length, 1)
        chanDim = -1
        if K.image_data_format() == "channels_first":
            inputShape = (1, freq_bins, length)
            chanDim = 1

        model.add(Conv2D(32, (3,3),
                         padding='same',
                         input_shape=inputShape,
                         kernel_initializer=keras.initializers.he_normal(seed=None),
                         bias_initializer='zeros'))
        model.add(Activation("relu"))
        model.add(BatchNormalization(axis=chanDim))
        model.add(Conv2D(32, (3,3),
                         padding='same',
                         kernel_initializer=keras.initializers.he_normal(seed=None),
                         bias_initializer='zeros'))
        model.add(Activation("relu"))
        model.add(BatchNormalization(axis=chanDim))
        model.add(MaxPooling2D(pool_size=(3,3)))
        model.add(Dropout(0.25))
```

```
model.add(Conv2D(64, (3, 3),
                 padding='same',
                 kernel_initializer=keras.initializers.he_normal(seed=None),
                 bias_initializer='zeros'))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(Conv2D(64, (3, 3),
                 padding='same',
                 kernel_initializer=keras.initializers.he_normal(seed=None),
                 bias_initializer='zeros'))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(3, 3)))
model.add(Dropout(0.25))


model.add(Conv2D(128, (3, 3),
                 padding='same',
                 kernel_initializer=keras.initializers.he_normal(seed=None),
                 bias_initializer='zeros'))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(Conv2D(128, (3, 3),
                 padding='same',
                 kernel_initializer=keras.initializers.he_normal(seed=None),
                 bias_initializer='zeros'))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(3, 3)))
model.add(Dropout(0.25))


model.add(Flatten())
model.add(Dense(freq_bins*2))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(freq_bins))
model.add(Activation('sigmoid'))
```

```
        model.compile(loss='mean_squared_error',
                      optimizer='adam')


        return model
```

Пакет generator.py

```
import numpy as np
import musdb
import librosa
import random
import keras
from time import time


class generator(keras.utils.Sequence):
    def __init__(self, steps_per_epoch, sbsts, splt, source_name, batch_size,
tracks_in_batch, sampling_rate, win_len, hop_len, sample_len):
        self.steps_per_epoch = steps_per_epoch
        self.tracks_in_batch = tracks_in_batch
        self.sbsts = sbsts
        self.splt = splt
        self.source_name = source_name
        self.batch_size = batch_size
        self.sampling_rate = sampling_rate
        self.win_len = win_len
        self.hop_len = hop_len
        self.sample_len = sample_len
        self.mus = musdb.DB(root='./musdb18', subsets=sbsts, split=splt)
        self.track_number = np.arange(len(self.mus))
        np.random.shuffle(self.track_number)
        self.cur_index = 0
        self.duration = (self.hop_len/self.sam-
pling_rate)*((self.batch_size/self.tracks_in_batch) + (self.sample_len-1)) +
(self.win_len-self.hop_len)/self.sampling_rate


    def __len__(self):
        return self.steps_per_epoch


    def __getitem__(self, index):
        mixes = []
        targets = []
        for i in range(self.tracks_in_batch):
            orig_sr, mix, target = self.get_random_track_piece()
            mix_power_spec = self.get_power_spec(mix,
                                                 orig_sr,
```

```
                                                self.sampling_rate,
                                                self.win_len,
                                                self.hop_len)
            target_power_spec = self.get_power_spec(target,
                                                    orig_sr,
                                                    self.sampling_rate,
                                                    self.win_len,
                                                    self.hop_len)
            mix_portion, target_portion = self.split_and_prep(mix_power_spec, tar-
get_power_spec)
            mixes = mixes + mix_portion
            targets = targets + target_portion
        # Преобразование списков в массивы
        mix_batch = np.array(mixes)
        target_batch = np.array(targets)
        # Добавление старшего измерения, соответствующего каналу
        mix_batch = np.expand_dims(mix_batch, len(mix_batch.shape))
        # Возврат батча в вызывающую функцию
        return mix_batch, target_batch

    def get_random_track_piece(self):
        random.seed(int(time()%1*1000000))
        if self.cur_index == self.track_number.shape[0]:
            np.random.shuffle(self.track_number)
            self.cur_index = 0
        track = self.mus[self.track_number[self.cur_index]]
        self.cur_index += 1
        track.chunk_duration = self.duration
        track.chunk_start = random.uniform(0, track.duration - track.chunk_duration)
        mix = track.audio.T
        target = track.targets[self.source_name].audio.T
        channel = random.randint(0, mix.shape[0]-1)
        return track.rate, mix[channel], target[channel]

    def get_power_spec(self, audio, orig_sr, new_sr, win_len, hop_len):
        if orig_sr!= new_sr:
            audio = librosa.core.resample(audio, orig_sr, new_sr)
        stft_ar = librosa.stft(audio, win_len, hop_len, center = False)
        return np.abs(stft_ar)

    def split_and_prep(self, mix_power_spec, target_power_spec):
        mix_portion = []
```

```
        target_portion = []
        for i in range(mix_power_spec.shape[1]-self.sample_len+1):
            mix_sample = np.copy(mix_power_spec[:,i:i+self.sample_len])
            target_sample = np.copy(target_power_spec[:,i+self.sample_len//2])
            target_sample = self.target_sample_to_mask(target_sample, mix_sample)
            target_portion.append(target_sample)
            mix_sample = self.normalize_pow_spec(mix_sample)
            mix_portion.append(mix_sample)
        return mix_portion, target_portion

    def target_sample_to_mask(self, target_sample, mix_sample):
        for f in range(target_sample.shape[0]):
            if mix_sample[f, self.sample_len//2] != 0:
                target_sample[f] /= mix_sample[f,self.sample_len//2]
                if target_sample[f] > 0.5:
                    target_sample[f] = 1
                else:
                    target_sample[f] = 0
        return target_sample

    def normalize_pow_spec(self, pow_spec):
        max = np.max(pow_spec)
        if max != 0:
            pow_spec /= max
        return pow_spec
```

Пакет train.py

```
import numpy as np
import librosa
import librosa.display
from keras.callbacks import ModelCheckpoint
from keras.models import load_model
from time import time
import matplotlib.pyplot as plt

from bss_cnn import BSS_CNN
from musdb_generator import generator

SR = 22050  # Частота дискретизации, с которой работает сеть
WIN = 1024  # Размер окна дискретного преобразования Фурье
HOP = 256    # Размер шага дискретного преобразования Фурье
```

```
TRGT = 'vocals' # Целевой источник, выделению которого будет обучатсья сеть
SMPL = 27   # Длина отрезков спектрограмм, подающихся на вход сети

BS = 64      # Размер мини-батча (пакета)
TIB = 16     # Количество разных музыкльных композиций в батче

EPOCHS = 3   # Количесво эпох обучения
SPE = 6500   # Количество шагов за эпоху (25% тренировочного набора)

print('<----------[INFO] batch generators creation...')
train_gen = generator(SPE, 'train', 'train', TRGT, BS, TIB, SR, WIN, HOP, SMPL)
valid_gen = generator(SPE, 'train', 'valid', TRGT, BS, TIB, SR, WIN, HOP, SMPL)

print('<----------[INFO] creation and compile network...')
model = BSS_CNN.define(freq_bins = int(WIN/2+1), length = SMPL)

print('<----------[INFO] training network...')
t0 = time()
H = model.fit_generator(generator=train_gen,
                        steps_per_epoch = SPE,
                        epochs = EPOCHS,
                        callbacks = [ ModelCheckpoint('output/model.hdf5', moni-
tor='val_loss', save_best_only=True, save_weights_only=False, mode='auto') ],
                        validation_data = valid_gen,
                        validation_steps = SPE,
                        max_queue_size = 20,
                        workers = 4
                        )
t1 = time()
print("<----------[INFO] model was trained in " + str(round((t1-t0)/60, 1)) + "
minutes")

print("<----------[INFO] evaluating network...")
# Построение графиков потерь на тренировочном и валидационном наборах
N = np.arange(1, EPOCHS+1)
plt.style.use('ggplot')
plt.figure()
plt.plot(N, H.history['loss'], label='train_loss')
plt.plot(N, H.history['val_loss'], label='val_loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Training Loss')
plt.legend()
```

```
plt.savefig('output/loss.png')

# Сохранение модели на диск
print("<----------[INFO] serializing network...")
model.save('output/cnn.hdf5')
```