

Санкт–Петербургский государственный университет

Балыкин Андрей Федорович

Выпускная квалификационная работа

*Методы интеллектуального анализа и генерации
звуковых сигналов*

Уровень образования: бакалавриат

Направление 01.03.02 «Прикладная математика и информатика»

Основная образовательная программа СВ.5005.2017 «Прикладная
математика, фундаментальная информатика и программирование»

Профиль «Математическое и программное обеспечение вычислительных
машин»

Научный руководитель:

доцент, кафедра технологии программирования,
к.ф. - м.н. Блеканов Иван Станиславович

Рецензент:

доцент, кафедра механики управляемого движения,
к.ф. - м.н. Шиманчук Дмитрий Викторович

Санкт-Петербург

2021 г.

Содержание

Введение	3
Актуальность работы	4
Цель работы	7
Задачи работы	7
Практическая значимость работы	8
Глава 1. Обзор методов цифровой обработки аудио сигналов	9
1.1. Дискретизация аудио сигнала	9
1.2. Спектр сигнала	12
1.3. Метрики громкости сигнала	13
1.4. Реверберация	15
Глава 2. Обзор методов интеллектуального анализа звуковых сигналов	16
2.1. Особенности нейронных сетей для анализа аудио	16
2.2. Wavenet	17
2.3. WaveGAN	20
2.4. SpecGAN	24
2.5. DDSP	25
Глава 3. Разработка программного комплекса для генерации звуков барабанов с использованием нейронных сетей	28
3.1. Архитектура программного комплекса	28
3.2. Подготовка авторского датасета	31
3.3. Выбор и обучение нейронных сетей	35
3.4. Анализ экспериментов	39
Заключение	41
Результаты работы	41
Перспективы развития	41
Список литературы	42

Введение

Каждый день нашей жизни мы сталкиваемся со звуками, которые слышим в окружающей нас реальности. С появлением новых технологий стало возможно записать эти звуки для того, чтобы в последствии использовать этот материал, например, для звукового оформления виртуальных пространств. Звуковое оформление встречается во многих сферах, начиная с киноиндустрии и театра, заканчивая производством компьютерных игр.

Для создания такого оформления требуются заранее подготовленные звуки, которые называются сэмплами. В настоящее время активно используются два подхода к их генерации: запись и синтез.

Запись звука производится с помощью микрофона в специально подготовленном помещении, при этом записываются звуки из физического пространства, в котором находится источник сигнала. Синтез производится путем генерации циклических волновых форм и их последующей обработки фильтром, огибающей громкости и эффектами. Эти подходы существуют с самого начала развития индустрии звукозаписи.

Новым подходом к созданию звуков является использование предобученных нейронных сетей для генерации аудио. С их помощью можно извлекать закономерности, которые присутствуют в определенном инструменте, голосе или выбранном множестве звуков, и с учетом этих особенностей генерировать новые аудиофрагменты, которые будут похожи на звуки из этого множества. Для обучения этих моделей требуется большое количество записанных или синтезированных звуков, и можно считать, что этот метод базируется на двух предыдущих подходах, дополняя и расширяя их возможности.

Модели глубоких нейронных сетей для генерации аудио в настоящее время только начинают свое развитие, и с каждым годом появляется все больше и больше прикладных проектов, решающих важные задачи из этой сферы. На подобные модели имеется большой спрос со стороны компаний производящих сэмплы, фильмы и видеоигры, поскольку аудио играет важную роль в этих индустриях.

Актуальность работы

В настоящее время машинное обучение и нейросетевые технологии активно развиваются и проникают во множество сфер нашей жизни. Разработчики музыкального программного обеспечения только начинают внедрять подобные подходы в свои продукты, но уже можно делать выводы об их эффективности и прикладной ценности.

С появлением сверточных архитектур нейронных сетей и продвинутых подходов к их обучению, применимость искусственного интеллекта в сфере анализа аудио существенно выросла. Это объясняется возросшим качеством анализа и генерации аудио с использованием таких моделей и накоплением существенного количества данных, которые можно анализировать.

Одной из первых таких работ в сфере генерации аудио, рассматриваемой в данной работе, является статья «WaveNet: A Generative Model For Raw Audio», где предлагается использовать сверточную сеть для условной генерации сигнала, что открывает новые возможности, например, построение Text-to-Speech модели [1].

Появившиеся позднее GAN сети показали улучшенное качество генерации по сравнению с более ранними архитектурами, поскольку сама связка генератор-дискриминатор лучше подходит под рассматриваемые задачи. Самыми популярными GAN сетями для задач, связанных с аудио, являются работы «GANSynth: Adversarial Neural Audio Synthesis» и «Adversarial Audio Synthesis» [2] [3]. Авторы последней работы также имплементировали программный комплекс похожий на тот, что разрабатывается в ходе данной работы, однако у них генерируются звуки низкого качества (16 кГц в секунду, генерация происходит в браузере), и в нем нельзя выбрать конкретный звук для генерации: звуки генерируются вперемешку.

Дополнительно, хочется отметить репозиторий Google Magenta включающий в себя множество Open Source проектов, работающих на нейросетях разработанных компанией Google [4]. Среди них можно отметить GanSynth для генерации и морфинга тембров живых или электронных инструментов, и набор инструментов DDSP для анализа и генерации аудио

сигналов.

«DDSP: Differentiable Digital Signal Processing» является одной из последних и актуальных работ по данной тематике, где преимущества классических инструментов цифровой обработки сигналов (Digital Signal Processing, DSP) сочетаются с методами глубокого обучения [5]. Такой подход дает гибкость при генерации и интерпретируемость при анализе результатов.

При наличии нужных тембров и нотной партитуры можно собрать полноценную аранжировку, эту задачу решает приложение Mubert, работающие на нейронной архитектуре и созданное для потоковой генерации музыки [6]. В нем доступно несколько жанров и степеней загруженности аранжировки, при этом весь трек целиком генерируется в режиме реального времени и доставляется на устройства пользователей путем передачи потокового аудио. При этом сгенерированная музыка может использоваться без лицензии и без отчисления роялти.

Нейронные сети также находят свое применение в плагинах для сведения аудио. Сведение – процесс микширования нескольких треков и обработки их различными эффектами для создания определенной звуковой картины, глубины и пространства музыкального произведения. Среди таких плагинов можно выделить эквалайзеры Soniebe Smart:EQ и Soundtheory Gullfoss [7], [8]. Для большей автоматизации процесса был разработан плагин Neutron Mixing Assistant от компании Izotope, который позволяет delegировать сведение интеллектуальной системе [9].



Рис. 1: Эквалайзер Soniebe Smart:EQ 3 [7].

Однако, наиболее удачно подобные технологии вписываются в сферу мастеринга музыкальных произведений. Мастеринг – процесс финальной доработки музыки, целью которого является получение необходимой для трека громкости, динамики и амплитудно-частотных характеристик. Среди таких продуктов можно выделить Izotope Ozone 9 Mastering Assistant и облачный сервис LANDR, которые производят мастеринг в автоматическом режиме [10] [11]. Также одним из инновационных продуктов является Izotope Ozone 9 Master Rebalance, который позволяет в уже готовом треке изменять баланс входящих в него инструментов.

Одной из проблем при работе с аудио материалом является необходимость использовать аналоговые приборы, поскольку они обладают эксклюзивными характеристиками. Устройство NeuralDSP Quad Cortex помогает в решении этих проблем путем эмуляции аналоговых приборов, обучая для каждого моделируемого устройства отдельную нейронную сеть, которая анализирует его характеристики и после этого эмулирует его поведение [12].

Из-за востребованности данных подходов, способы применения таких технологий только начинают активно разрабатываться, и в будущем мы увидим множество других возможных приложений новейших моделей к музыкальным задачам, одно из которых рассматривается в данной работе.

Цель работы

Цель данной работы состоит в разработке программного комплекса для генерации различных категорий звуков. В качестве специфики рассматривается генерация звуков барабанной установки, где для каждого барабана обучается отдельная нейронная сеть с необходимыми для этого параметрами. После генерации для каждого барабана реализуется своя постобработка сгенерированного аудио, и из полученных для каждого барабана по отдельности сэмлов можно собрать барабанную партию, прописав для каждого из этих звуков ритмический паттерн. Для построения такого программного комплекса требуется изучение возможностей использования современных нейросетевых технологий в задаче анализа и генерации звукового сигнала.

Генерируя звуки важно правильно передать тембр звука, анализируя закономерности при обучении выбранной модели и минимизируя при этом искажения, возникающие в результате генерации. Одним из желаемых результатов является получение звуков с необходимым для производства музыкального контента качеством, которое обеспечивается высокой частотой дискретизации, отсутствием артефактов и шумов.

Задачи работы

В соответствии с целями данной работы был поставлен следующий список задач:

- Провести обзор методов цифровой обработки и интеллектуального анализа аудио сигналов
- Подготовить авторский датасет
- Обучить выбранную модель, провести анализ сгенерированных звуков, исследовать возможности их постпроцессинга
- Разработать программный комплекс для генерации звуков в цифровой рабочей станции (Digital Audio Workstation, DAW)

Практическая значимость работы

Развитие индустрии аудио в последние годы не обошло стороной и продажу звуков, появилось множество агрегаторов, имеющих большой каталог сэмплов. Среди них можно выделить сервис Splice Sounds пользующийся наибольшей популярностью [13]. Используя данный сервис, можно найти интересующие пользователя звуки из большого количества категорий, которые включают в себя звуки барабанов и готовые барабанные партии.

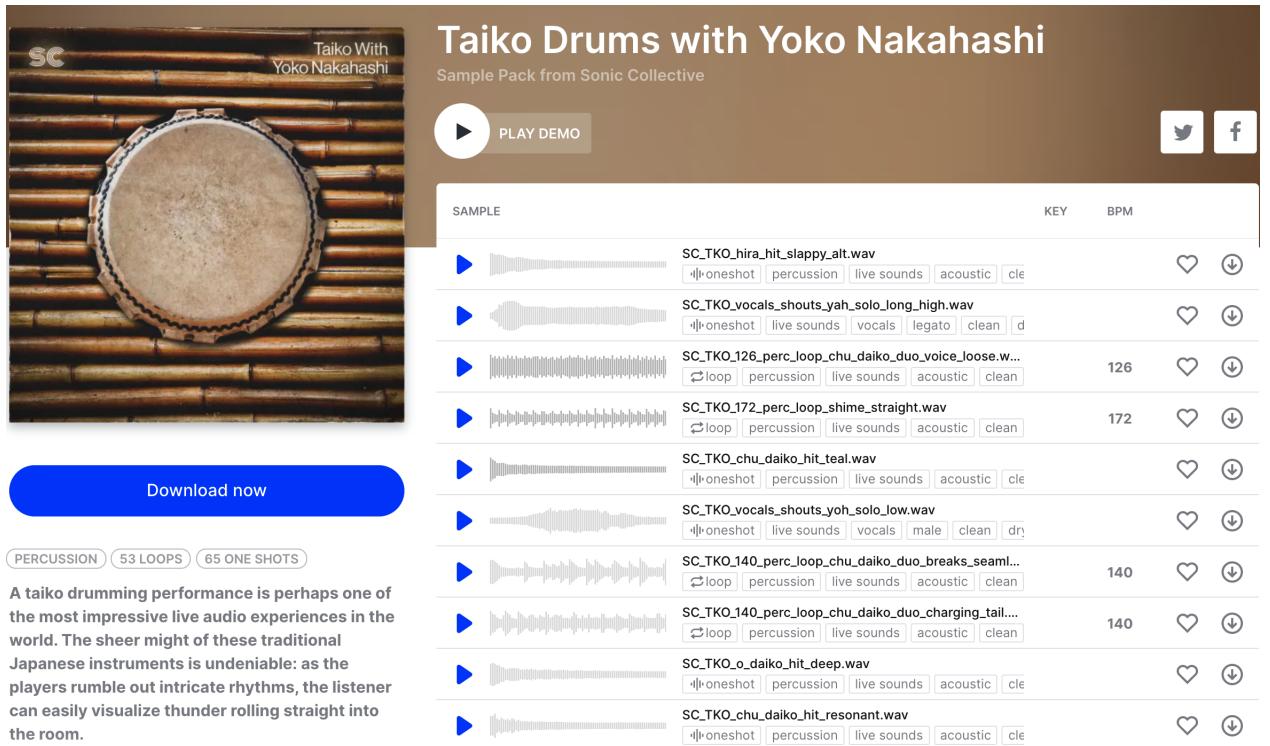


Рис. 2: Сервис Splice Sounds [13].

Данная работа рассматривает способ генерации аудио с помощью нейронных сетей, как альтернативу подобным сервисам для некоторых категорий барабанных звуков. Сгенерированные таким образом звуки можно свободно применять для производства различных музыкальных композиций или построения генеративных музыкальных моделей.

Глава 1. Обзор методов цифровой обработки аудио сигналов

1.1 Дискретизация аудио сигнала

В окружающем нас мире звук имеет колебательную и непрерывную природу, и в случае использования аналоговых устройств, таких как микрофоны, звукозаписывающие консоли или магнитные ленты, эта природа сохраняется, так как для электрического тока и магнитных полей тоже выполняется свойство непрерывности. Однако для представления таких данных в памяти компьютера необходимо перейти от непрерывных уровней к дискретным, этот процесс называется дискретизацией.

Процесс перехода от непрерывного звука к дискретному происходит с помощью специальных устройств: Аналогово-Цифрового и Цифро-Аналогового преобразователя (АЦП и ЦАП). Такие устройства используются повсеместно начиная от звуковых карт, заканчивая портативными устройствами. Для дискретизации сигнала выбираются следующие параметры:

- Частота дискретизации — количество измерений уровня входного сигнала аналогово-цифровым преобразователем за одну секунду.
- Глубина дискретизации — количество возможных уровней для каждого сэмпла.

Глубина дискретизации по уровню задает разрядность числа, которым описывается уровень сигнала. Эта характеристика не имеет отношения к громкости — она накладывает ограничение на точность записи уровня сигнала и на его минимальное значение. Если известна глубина дискретизации, то количество уровней громкости цифрового звука можно рассчитать по формуле $N = 2^i$. Если глубина дискретизации звука составляет 16 битов, тогда количество уровней громкости звука равно: $N = 2^i = 2^{16} = 65536$.

Стандартная глубина дискретизации на audio-CD — 16 бит. При этом, если не использовать специальную студийную аппаратуру, разницу в звучании большинство перестаёт замечать уже в районе 10-12 бит. Однако большая глубина дискретизации позволяет избежать появления шумов при дальнейшей обработке звука и увеличивает динамический диапазон.

Самыми распространёнными для звукозаписи являются частоты дискретизации: 44.1 (CD качество), 48 (DVD стандарт) и 92 кГц. Частота дискретизации 16 кГц не подходит для профессиональной звукозаписи, но так как она используется в радиовещании и телефонии, она свое прикладное значение и большинство нейронных моделей обучают на это частоте в целях экономии ресурсов [14].

Существует множество способов кодирования, но самым распространённым из них является импульсно-кодовая модуляция, при которой записывается не конкретное значение уровня сигнала в каждый момент времени, а разница между текущим и предыдущим значением. Это позволяет снизить количество бит на каждый отсчёт примерно на 25%. Этот способ кодирования применяется в наиболее распространённых аудио-форматах (WAV, MP3, OGG, FLAC), которые используют контейнер PCM WAV [15].

В реальности для создания стереоэффекта при записи аудио чаще всего записывается не один, а сразу несколько каналов. В зависимости от используемого формата они могут храниться, как независимо, так и как разница между уровнем основного канала и уровнем текущего.

Теорема Найквеста-Котельникова

- Любой аналоговый сигнал может быть восстановлен с какой угодно точностью по своим дискретным отсчётам, взятым с частотой $f > 2f_c$, где f_c — максимальная частота, которая ограничена спектром реального сигнала;
- Если максимальная частота в сигнале равна или превышает половину частоты дискретизации (наложение спектра), то способа восстановить сигнал из дискретного в аналоговый без искажений не существует.

Пользуясь следствиями из этой теоремы, можно заключить, что частота дискретизации 44100 позволяет работать в частотном диапазоне до 22кГц, 48000 до 24кГц, 96000 до 48кГц. Этим объясняется тот факт, что частота 16 кГц не используется при звукозаписи, поскольку ее частотный диапазон ограничен 8 кГц, при том, что человеческий слух в среднем слышит частоты до 20 кГц, то есть высокочастотная составляющая звука теряется.

Традиционно, в цифровой звукозаписи аудиодорожка представляется в виде осциллографии, отображающей форму звуковой волны (waveform), то есть зависимость амплитуды звука от времени. Такое представление достаточно наглядно: осциллограф позволяет увидеть основные события в звуке, такие как изменения громкости, паузы между частями произведения и зачастую даже отдельные ноты в сольной записи инструмента (рис. 2). Но одновременное звучание нескольких инструментов на осциллографии смешивается и визуальный анализ сигнала становится затруднительным. Для анализа сложных сигналов применяются спектроанализаторы.



Рис. 3: Пример волновой формы записанной барабанной партии.

1.2 Спектр сигнала

Спектроанализатор – прибор для измерения и отображения спектра сигнала – распределения энергии сигнала по частотам. Современные анализаторы основаны на FFT – быстром преобразовании Фурье. При работе со звуком используется анализ конечных по времени звуков, и поскольку музыка не является статичным сигналом, её спектр меняется во времени. Поэтому при спектральном анализе нас обычно интересуют отдельные короткие фрагменты сигнала. Для анализа таких фрагментов цифрового сигнала существует дискретное преобразование Фурье:

$$x(n) = \sum_{k=0}^{N/2} X_k \cos\left(\frac{2\pi k(n + \phi_k)}{N}\right)$$

В этой формуле N отсчётов дискретного сигнала $x(n)$ на интервале времени от 0 до $N-1$ синтезируются как сумма конечного числа синусоидальных колебаний с амплитудами X_k и фазами ϕ_k . Частоты этих синусоид равны kF/N , где F – частота дискретизации сигнала, а N – число отсчётов исходного сигнала $x(n)$ на анализируемом интервале. Набор коэффициентов X_k называется амплитудным спектром сигнала.

FFT (Fast Fourier transform) – алгоритм быстрого вычисления дискретного преобразования Фурье. Благодаря ему стало возможным анализировать спектр звуковых сигналов в реальном времени и использовать спектроанализаторы [17].

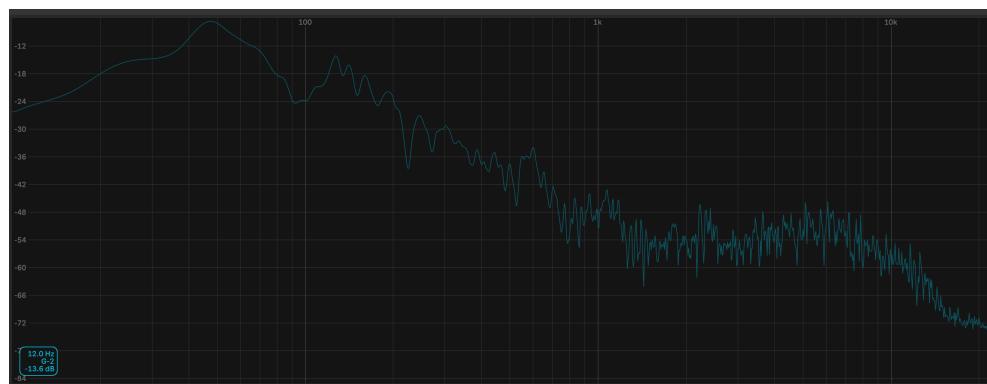


Рис. 4: Спектр части волновой формы из рис.3.

1.3 Метрики громкости сигнала

При работе с цифровыми сигналами можно говорить только об относительном уровне сигнала. Абсолютный зависит в первую очередь от воспроизводящей аппаратуры и прямо пропорционален относительному. При расчётах относительных уровней сигнала принято использовать децибелы. При этом за точку отсчёта берётся сигнал с максимально возможной амплитудой при заданной глубине дискретизации. Этот уровень указывается как 0 dBFS (dB — децибел, FS = Full Scale — полная шкала). Более низкие уровни сигнала указываются как -1 dBFS, -2 dBFS и т.д. Каждые $6.02dB = 20 \log_{10} 2$ указывают на изменение реального уровня сигнала в два раза. Поэтому сигнал с уровнем -12.04 dBFS, понимается, как сигнал, уровень которого в четыре раза меньше максимального, а с уровнем -18.06 dBFS — в восемь.

Также одним из способов измерения уровня сигнала за какой-либо период времени является RMS измерение [18]. Измерение RMS (Root Mean Square) показывает среднеквадратичный выходной уровень, рассчитанный в течение короткого периода времени. RMS чаще всего будет отображаться ниже эквивалентного пикового метра, поскольку пики усредняются в общей громкости, что дает более стабильный и непрерывный уровень, который ближе к нашему восприятию по сравнению с быстро меняющимся уровнем.

Однако, в отличие от человеческого уха, которое имеет различную чувствительность на разных частотах (мы воспринимаем изменения громкости на средних частотах гораздо более остро, чем на низких и высоких), измерители уровней приведенные выше, не могут различать частотные диапазоны. Чтобы учесть это, была создана совершенно новая шкала измерения, называемая LUFS (Loudness Units Full Scale) [18].

Шкала LUFS широко используется в индустрии вещательного телевидения, видеоиграх и стриминговых сервисах. Некоторые игровые компании указывают, что музыкальный трек должен находиться в пределах определенных уровней LUFS, чтобы они могли лучше предсказать, как он будет балансироваться с другими звуками игры.

Значения данных измерителей можно использовать для оценки сгенерированных звуков, так как при обучении какой-либо из архитектур нейронных сетей, ожидается, что выходные данные будут иметь значения громкости близкие к значениям из обучающей выборки.



Рис. 5: Специализированное измерительное ПО IZotope Insight [19].

1.4 Реверберация

Также в некоторых звуках, анализ которых будет проводится, существует эффект реверберации. Реверберация – постепенное уменьшение интенсивности звука, сопровождающееся появлением множества эхо при его многократных отражениях в замкнутом пространстве. При записи звука близко к микрофону влияние комнаты на запись уменьшается и звук записывается без слышимого эффекта его нахождения в комнате. В современной студийной практике реверберацию используют для того, чтобы алгоритмически создать эффект нахождения звука в каком-либо пространстве. Так или иначе какое-то количество звуков из датасета окажется записано с реверберационным хвостом, поэтому сеть так или иначе будет обучаться эмулировать реверберацию при генерации звуков.

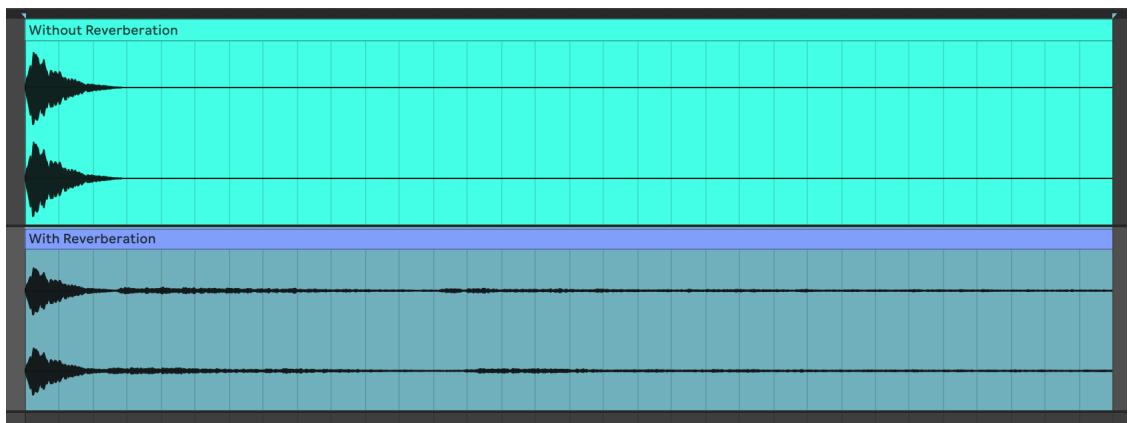


Рис. 6: Звук до и после реверберации.

Глава 2. Обзор методов интеллектуального анализа звуковых сигналов

2.1 Особенности нейронных сетей для анализа аудио

В настоящее время большинство архитектур нейронных сетей заточены под задачи классификации, сегментирования или генерации изображений, поскольку эти сферы являются наиболее приоритетными. После получения положительных результатов на изображениях, модель пытаются адаптировать для использования на аудио данных.

Интеллектуальный анализ цифровых аудио сигналов отличается от анализа изображений, для которых изначально были разработаны сети, и при построении таких сетей для обработки звука важно учитывать эти отличия. Одним из вариантов для иллюстрации отличий между аудио и изображениями является анализ главных компонент.

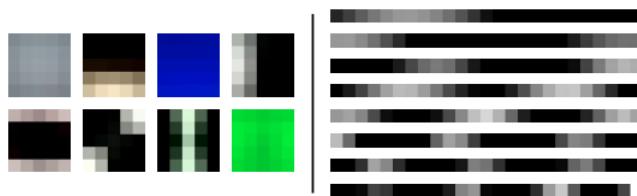


Рис. 7: Сравнение главных компонент изображений и звуков [3].

На рисунке 7 изображены главные компоненты нескольких изображений и нарезки речи. В то время как основные компоненты изображений обычно захватывают характеристики интенсивности, градиента и краев объектов, компоненты звука образуют периодические паттерны, которые определяют составляющие частоты. Как правило, аудио сигналы с большей вероятностью демонстрируют периодичность, чем изображения.

Как следствие, корреляция между большими промежутками отсчетов есть регулярное явление в аудио. Например, для формы волны синусоиды 440 Гц (нота А4), при дискретизации с частотой 16 кГц для завершения одного цикла требуется более 36 отсчетов. Это говорит о том, что модели с большими восприимчивыми слоями необходимы для обработки звуковых данных.

2.2 Wavenet

WaveNet – авторегрессионная вероятностная модель, созданная подразделением Deep Mind компании Google для генерации text-to-speech и других аудио сигналов [1]. Данная сеть является адаптацией сети PixelCNN к задаче генерации аудио и одновременно самым ранним среди методов, рассматриваемых в данной работе.

В качестве отправной точки для построения модели была выбрана вероятность появления волновой формы $x = \{x_1, x_2, \dots, x_T\}$ на выходе сети, которая равна произведению вероятностей появления сэмпла x_t при условии появления x_1, \dots, x_{t-1} . Таким образом каждый следующий сэмпл аудио зависит от всех предыдущих сэмплов.

$$p(x) = \prod_{t=1}^T p(x_t | x_1, \dots, x_{t-1})$$

Условное вероятностное распределение моделируется с помощью сверточных слоев, среди которых нет пулинговых. Это было сделано для того, чтобы размерность входного и выходного сигнала сети совпадала. Каждый сэмпл на выходе попадает под категориальное распределение (то есть это в некотором роде классификатор громкости каждого сэмпла) с softmax слоем, данная модель оптимизирует логарифмическую функцию правдоподобия. В виду того, что 16 бит аудио имеет 65536 вариантов громкости сэмпла, классов становится слишком много. Для решения этой проблемы уменьшается количество классов с помощью применения мю-закона [20]. Таким образом уровень квантуется к 256 возможным значениям по формуле:

$$f(x_t) = sign(x_t) \frac{\ln(1 + \mu|x_t|)}{\ln(1 + \mu)}$$

где $-1 < x_t < 1$ и $\mu = 255$.

Одной из главных особенностей WaveNet является причинная свертка. С ее помощью мы можем гарантировать, что сеть не нарушит порядок при моделировании распределения, то есть шаг t_n может зависеть только от шагов t_1, \dots, t_{n-1} , но не от шагов t_{n+1}, \dots, t_k (будущее). Поэтому обучение модели может выполняться параллельно (т.к. данные уже известны), а генерация выполняется последовательно поскольку каждый следующий сэмпл влияет на все будущие сэмплы. Так как в данной модели генерируются одномерные данные, свертка в данном случае тоже одномерная, поэтому она имеет визуальные и смысловые отличия от привычной двумерной свертки, использующейся при анализе изображений.

Также большим преимуществом данной модели является возможность выделять закономерности, которые проявляются через сотни сэмплов, что очень важно при генерации периодических сигналов. Это достигается с использованием расширенной свертки (свертки с пропусками), которая пропускает некоторые отсчеты сигнала, воспринимая более длительные закономерности.

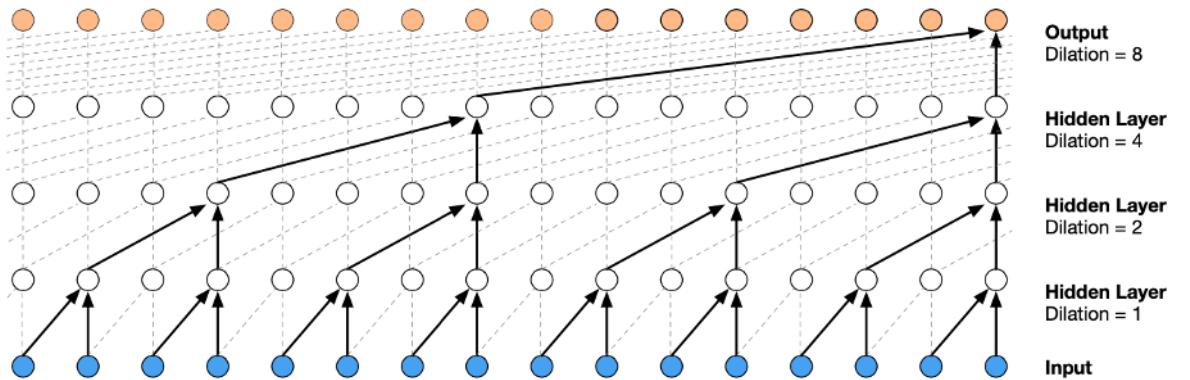


Рис. 8: Пример расширенной свертки [1].

Множество таких слоев позволяет воспринимать значительно длинные участки аудио. В данной модели применяются смещения рассчитанные по степеням двойки: 1, 2, 4, 8, 16, ..., 512.

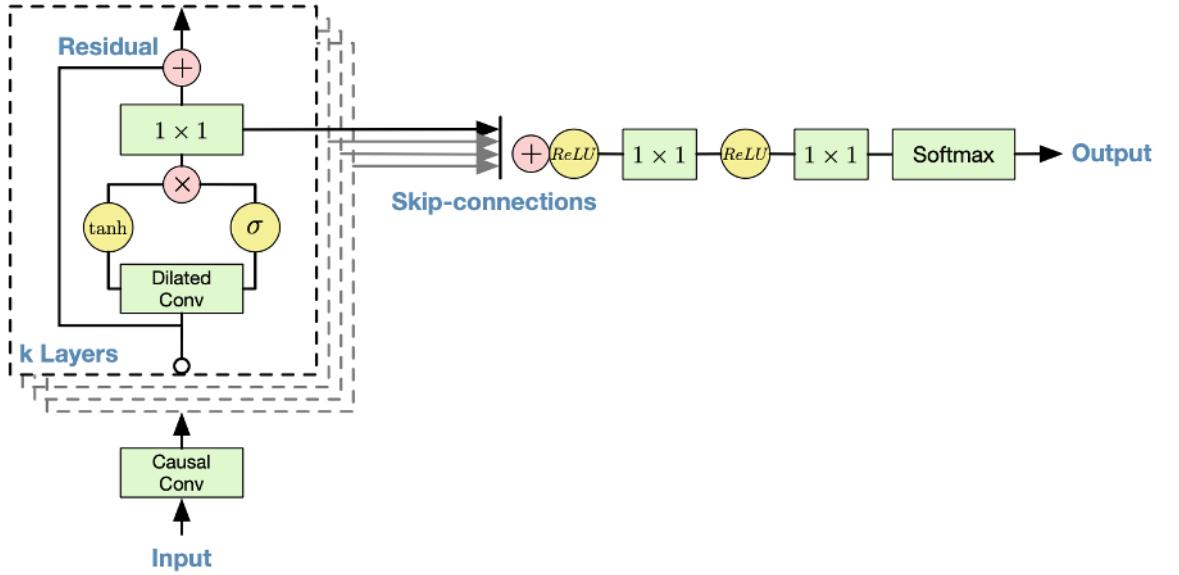


Рис. 9: Архитектура сети WaveNet [1].

В рассматриваемой модели также имплементирована условная генерация, которая позволяет сделать так, чтобы выходной сэмпл зависел не только от предыдущих сэмлов, но и от некоторого множества параметров h :

$$p(x) = \prod_{t=1}^T p(x_t | x_1, \dots, x_{t-1} | h)$$

Этим множеством может выступать строка, поэтому одним из важных приложений WaveNet является Text-To-Speech генерация, такой результат достигается тем, что к входному сигналу добавляют букву или слово, которое произносится в аудиофрагменте, после этого обученная сеть может генерировать аудио из текста.

Из минусов данной модели можно отметить наиболее продолжительное среди остальных сетей время обучения до приемлемых результатов. Также качество генерации аудио у GAN в большинстве случаев выше.

2.3 WaveGAN

Целью обучения GAN сетей является отображение скрытого вектора $z \in Z$ в пространство тренировочных данных X . В первоначальном варианте генератор $G : Z \implies X$ противопоставлялся дискриминатору $D : X \implies [0, 1]$ в игре с двумя игроками с использованием минимакс стратегии [21]. Целью обучения генератора является минимизация приведенной ниже функции, а целью дискриминатора ее максимизация:

$$V(D, G) = E_{x \sim P_X}[\log D(x)] + E_{z \sim P_Z}[\log(1 - D(G(z)))]$$

Другими словами, дискриминатор обучается для того, чтобы отличать настоящие данные от сгенерированных, в то время как генератор пытается подделать сгенерированные данные максимально похоже на настоящие. Предложенный для обучения алгоритм минимизирует дивергенцию Йенсена — Шеннона между распределениями исходных данных P_X и сгенерированных P_G [21]. В первоначальной формулировке GAN сеть сложно обучалась и показывала нестабильные результаты. Позже подход был усовершенствован, и вместо дивергенции Йенсена — Шеннона в [22] была предложена минимизация более гладкого расстояния Вассерштейна:

$$W(P_X, P_G) = \sup_{\|f\|_L \leq 1} E_{x \sim P_X}[f(x)] - E_{x \sim P_G}[f(x)]$$

где $\|f\|_L \leq 1 : X \implies R$ семейство Липшицевых функций первого порядка.

Для минимизации этого расстояния был предложен алгоритм WGAN, который минимизирует функцию похожую на первоначальную:

$$V_{WGAN}(P_X, P_G) = E_{x \sim P_X}[D_w(x)] + E_{z \sim P_Z}[D_w(G(z))]$$

Но в этой формулировке D_w (дискриминатор) обучается не отличать сгенерированные примеры от настоящих, а вместо этого обучается как функция, помогающая вычислять расстояние Вассерштейна. Используя идею штрафа за отклонения градиента от единичной нормы (gradient penalty) была имплементирована следующая версия сети WGAN-GP, которой получалось обучаться в тех случаях, когда остальным архитектуркам GAN сети этого не удавалось.

WaveGAN - созданная Chris Donahue и Julian McAuley сеть, базирующаяся на архитектуре Deep Convolutional Generative Adversarial Network (DCGAN), используемой в основном для обучения на изображениях [3].

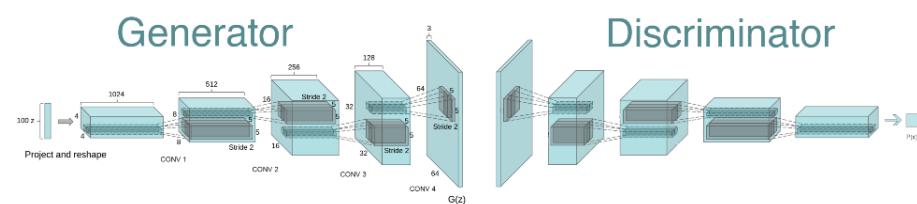


Рис. 10: Архитектура DCGAN [24].

Генератор DCGAN данной сети использует транспонированную свертку для того, чтобы повышать размерность небольших матриц скрытых признаков в изображение высокой четкости (upsampling). В отличии от DCGAN в WaveGAN вместо свертки 5x5 используется одномерная свертка размерности 25x1, и размерность upsampling'a была увеличена в два раза.



Рис. 11: Сравнение свертки для изображений и аудио [3].

Дискриминатор использует тот же вид свертки, что и генератор, только вместо увеличенного в два раза upsampling'a используется увеличенный в два раза шаг свертки. В результате проделанной замены сеть имеет такое же количество параметров и выходных размерностей, что и DCGAN.

Сеть DCGAN имеет выход 64×64 , что в одномерном случае равно 4096 сэмплам, поэтому в WaveGAN был добавлен еще один слой превращающий выход из 4096 в 16384 сэмплов, что есть чуть больше, чем одна секунда аудио при частоте дискретизации 16 кГц, чего вполне достаточно для коротких звуков. Спустя год после выхода статьи вышла новая версия WaveGAN с возможностью генерировать аудио с произвольной частотой дискретизации до 4 секунд на частоте дискретизации 16 кГц и до одной секунды на частоте 44.1 кГц.

Основные модификации, которые были сделаны над сетью DCGAN для работы со звуком:

- Двумерная свертка спрятана в одномерную, то есть 5×5 свертка из DCGAN стала одномерной сверткой с 25 компонентами.
- Увеличен шаг для всех сверток, т.е. шаг 2×2 в двумерном виде стал шагом 4 в одномерном.
- Из генератора и дискриминатора убрана батч нормализация.
- Обучение производится используя алгоритм WGAN-GP.

При генерации изображений с помощью DCGAN может возникнуть так называемый «эффект шахматной доски», который обусловлен построением генератора этой модели [23]. При генерации изображений периодическая информация встречается редко, поэтом дискриминатору легко отличать реальные изображения от сгенерированных.



Рис. 12: Эффект шахматной доски [23].

Эффект шахматной доски в аналогии с аудио воспринимается, как шум с постоянной частотой, но в отличии от изображений дискриминатору должно быть намного тяжелее его распознать в контексте аудио. Однако это шум всегда возникает с определенной фазой и дискриминатор может подобрать тривиальный алгоритм по которому он будет судить о подлинности аудиофайла. Для того чтобы дискриминатор не смог воспользоваться этим недостатком генератора применяется перемешивание фазы с гиперпараметром n : этот алгоритм случайным образом нарушает фазу каждого слоя активаций от $-n$ до n сэмплов перед выходом на следующий слой. Перемешивание фазы применяется только для дискриминатора, поскольку скрытое векторное представление изначально дает генератору механизм для управления фазой результирующего сигнала.

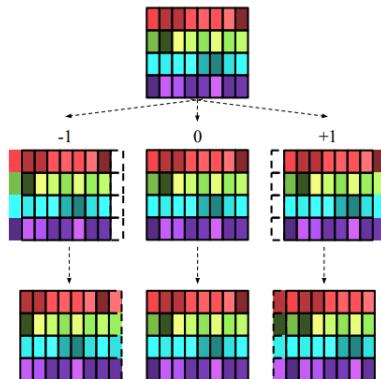


Рис. 13: Демонстрация перемешивания фазы при $n = 1$ [3].

2.4 SpecGAN

В отличии от задач генерации, в задачах классификации аудио преобладает использование спектрограмм, а не волновых форм. Генеративная модель также может выиграть от работы в спектровременном пространстве. Однако переход от волновой формы к спектру в том виде, в каком он присутствует в этих моделях в общем случае не обратим.

Модель SpecGAN является альтернативой модели WaveGAN, которая обучается на спектрограммах вместо волновых форм. В отличии от 16384 сэмплов в случае WaveGAN, предложенная модель генерирует двумерные 128x128 спектрограммы, которые можно трактовать, как изображения [3].

Чтобы преобразовать звук в подходящие спектрограммы, выполняется быстрое преобразование Фурье с окном 16 мс и шагом 8 мс, в результате получается 128 интервалов частот линейно разнесенных от 0 до 8 кГц. Далее амплитуда спектра логарифмируется для соответствия человеческому восприятию звука и каждый из интервалов нормализуется.

Чтобы отобразить результирующие сгенерированные спектрограммы в виде аудио сигналов, сначала инвертируются шаги предварительной обработки спектрограммы, описанные выше, в результате чего получаем линейно-амплитудную частотную характеристику. Затем используется итерационный алгоритм Гриффина-Лима с 16 итерациями для оценки фазы и создания 16384 аудиосэмплов.

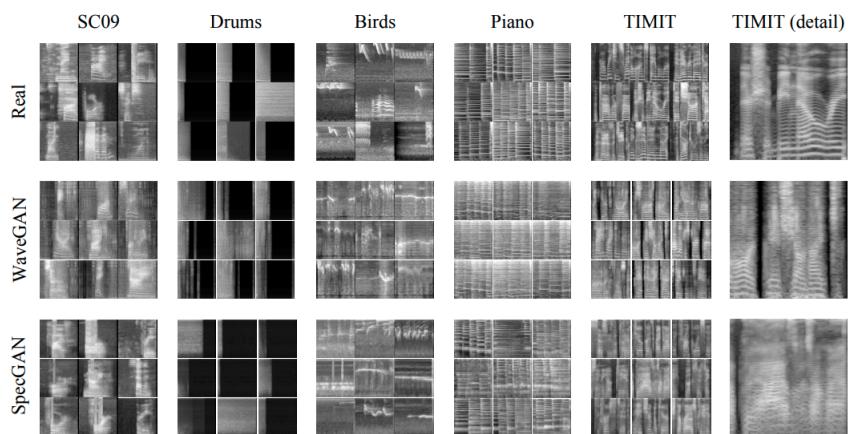


Рис. 14: Примеры датасетов и выходных данных сетей WaveGAN и SpecGAN [3].

2.5 DDSP

Библиотека DDSP обеспечивает прямую интеграцию классических элементов обработки цифровых сигналов с методами глубокого обучения. В оригинальной статье она противопоставляется таким подходам, как сверточные и авторегрессионной модели, фокусируясь на синтезе звука с генерацией высокой точности [5]. DDSP обеспечивает интерпретируемый и модульный подход к генеративному моделированию без утраты преимуществ глубокого обучения.

В качестве генератора в модели используется синусоидальный осциллятор, выход которого может описан как:

$$x(n) = \sum_{k=1}^K A_k(n) \sin(\phi_k(n))$$

где $A_k(n)$ меняющаяся со временем амплитудная характеристика k – синусоидальной компоненты, $\phi_k(n)$ - их мгновенная фаза, которая получается с помощью суммирования мгновенных чатот $f_k(n)$ и начальной фазы $\phi_{0,k}$, которая может задаваться случайно или определяться при обучении модели:

$$\phi_k(n) = 2\pi \sum_{m=0}^n f_k(m) + \phi_{0,k}$$

Для гармонического осциллятора все синусоидальные частоты являются гармоническими, т.е. кратными основной частоте $f_0(n)$: $f_k(n) = k f_0(n)$. Таким образом, выход гармонического осциллятора полностью параметризуется изменяющейся во времени основной частотой $f_0(n)$ и амплитудами гармоник $A_k(n)$.

$$A_k(n) = A(n) c_k(n)$$

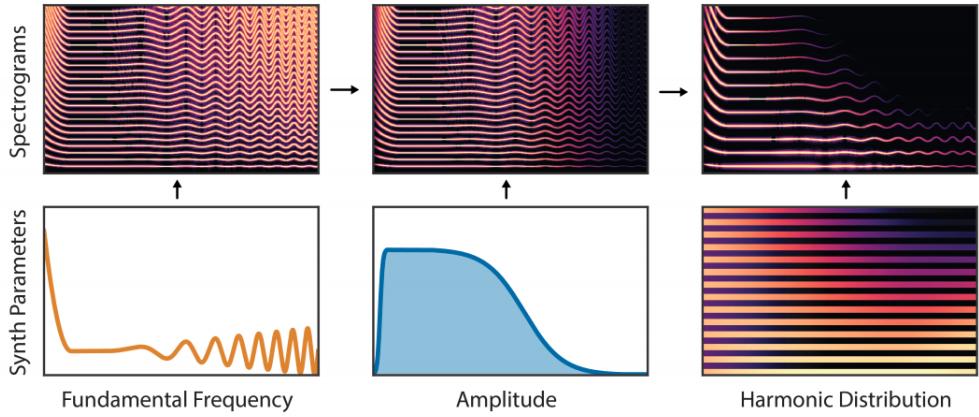


Рис. 15: Диаграмма работы гармонического осциллятора [5].

В звуках с которыми мы регулярно сталкиваемся присутствуют не только гармонические, но и стохастические компоненты, поэтому для построения более гибкой модели к гармоническому осциллятору добавляется фильтрованный шум, что позволяет получать больший диапазон звуков.

Модель DDSP является сочетанием методов цифровой обработки сигналов и глубокого обучения и состоит из двух частей (рис 16). Левая часть представляет собой автоэнкодер, который определяет фундаментальную частоту, громкость и слой скрытых представлений, направляя полученные параметры в правый блок. Он отвечает за генерацию аудио и последующую его обработку сверточным ревербератором. Красным цветом на рисунке показываются компоненты, где используются глубокие нейронные сети, а желтым обозначены DSP компоненты.

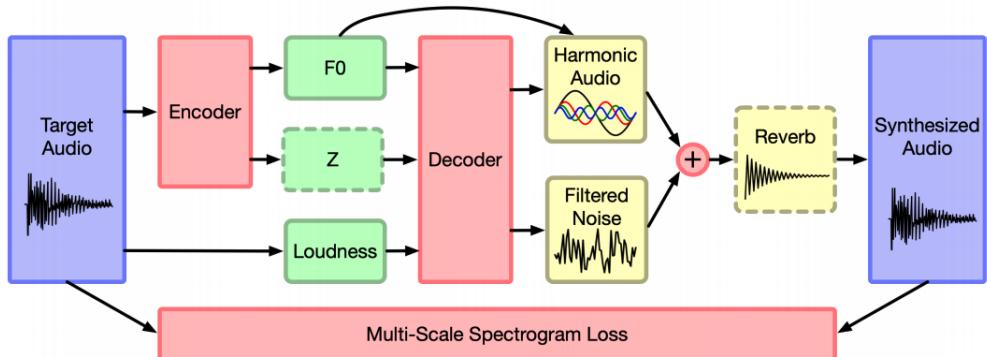


Рис. 16: Архитектура сети DDSP [5].

Интерпретируемость и модульность данной сети позволяет :

- Независимо контролировать громкость и высоту тона в процессе генерации.
- Реалистично экстраполировать результаты на те высоты тона, которых не было в обучающей выборке.
- Производить дереверберацию полученного тембра, так как за реверберационный процесс отвечает отдельный модуль в конце сети.
- Производить трансфер из одного акустического пространства в другое.
- Производить трансфер тембра партии одного инструмента на партию другого инструмента.

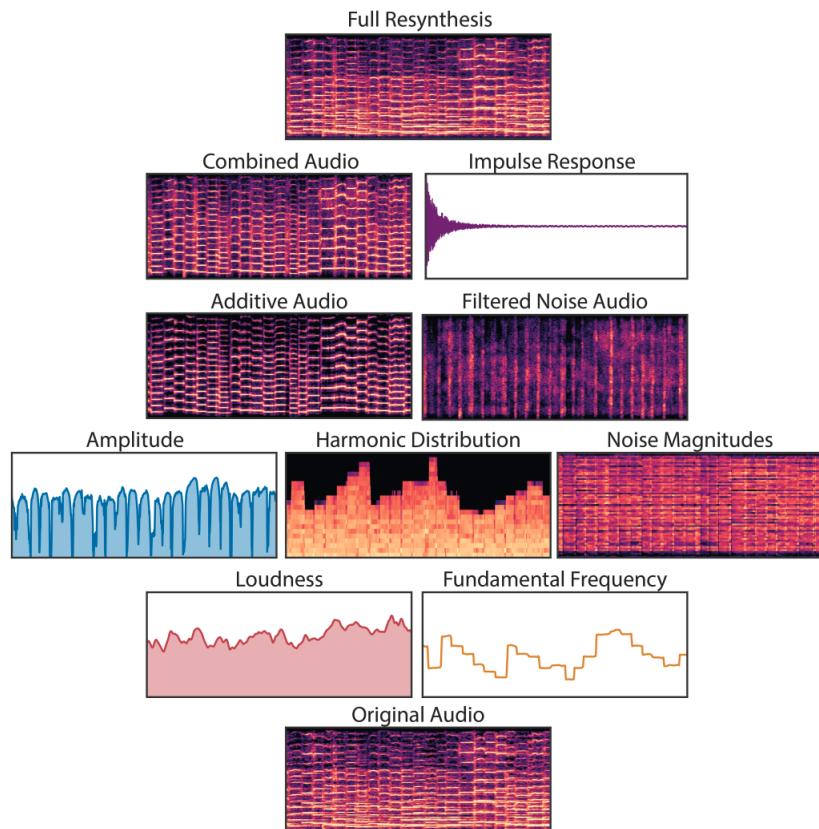


Рис. 17: Схема работы DDSP [5].

Глава 3. Разработка программного комплекса для генерации звуков барабанов с использованием нейронных сетей

3.1 Архитектура программного комплекса

Для имплементирования возможности генерировать звуки без существенной нагрузки на персональный компьютер пользователя было принято решение разработать программный комплекс базирующийся на клиент-серверной архитектуре [25].

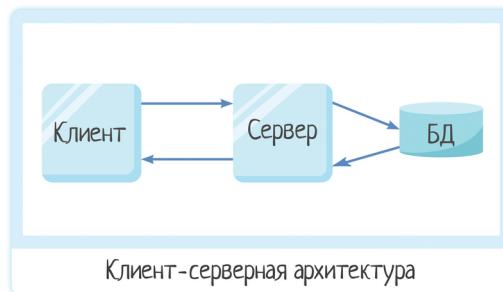


Рис. 18: Клиент-Серверная Архитектура [25].

Сервер в данном случае отвечает за генерацию, постобработку, и хранение, как самих звуков, так и чекпоинтов для их генерации. Так как данный программный комплекс разрабатывается в качестве прототипа, в нем отсутствует параллельная генерация и другие возможные механизмы по балансировке нагрузки на сервер. Однако такие подходы могут и не потребоваться, если генерировать каждый день большое количество звуков и случайным образом отдавать их каждому пользователю.

Клиент-приложение используется для встраивания сгенерированных звуков в цифровую рабочую станцию, с помощью которой можно применять их в студийной работе. Приложение должно уметь общаться с сервером, загружать сгенерированные звуки, ставить их на определенные ноты и воспроизводить звук при нажатии на ноту. Опционально, в подобном приложении может использоваться дополнительный DSP процессинг сгенерированных звуков.

Для применения обученных моделей на практике был разработан программный комплекс, построенный на клиент-сервер архитектуре. Для взаимодействия клиентской части с сервером был разработан API на языке программирования python с использованием библиотек flask, и flask-restful. Данные библиотеки позволяют использовать готовые классы для обработки запросов, получения параметров и отправке ответов на эти запросы, что значительно ускоряет разработку серверной части.

При HTTP запросе к серверу из клиент-приложения указываются параметры для генерации: тип инструмента, метод и параметры интерполяции, если она необходима. При получении запроса происходит развертывание чекпоинта требуемого инструмента, генерация звука моделью и последующая интерполяция выбранным методом с использованием переданных для нее параметров. Далее звук отправляется на приложение-клиент как ответ на запрос к серверу.

Развертывание чекпоинта и генерация звуков происходит почти мгновенно, что дает пользователю возможность генерировать звуки почти в “реальном времени” и положительно влияет на пользовательский опыт взаимодействия с данным программным комплексом. Однако в случае большой нагрузки на сервер, например, при обращении к нему одновременно большого количества запросов однопоточный сервер не сможет корректно их отработать. Обычно для решения данной проблемы применяют распределение нагрузки по нескольким серверам или потокам, но в случае данного прототипа это было бы излишним. В связи с этим было принято решение ежедневно генерировать большую выборку звуков, которые случайным образом будут выбираться и оправляться в ответ на запросы, таким образом каждый пользователь сможет получить необходимое количество звуков без существенной нагрузки на сервер.

Клиентская часть данного программного комплекса реализована в среде программирования Max 8 с использованием Max/MSP библиотеки. Выбор данной среды объясняется тем, что с ее помощью можно разрабатывать плагины для цифровой рабочей станции Ableton Live.

В разработанной клиентской части приложения можно выбрать категорию инструмента, которую необходимо сгенерировать, включить или отключить сглаживание волновой формы. Также в программе реализовано визуальное отображение волновой формы звука.

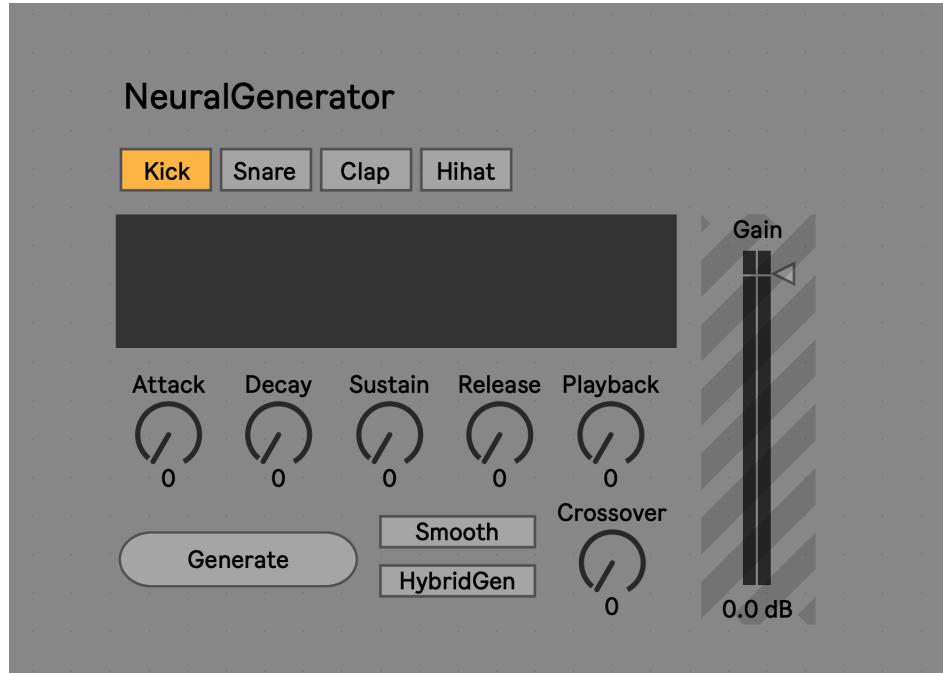


Рис. 19: Интерфейс клиентской части программного комплекса.

Для избавления от искажений в концах звуков применяется ADSR огибающая громкости, которую можно настроить с помощью ручек интерфейса (Attack, Decay, Sustain, Release). Скоростью воспроизведения звука можно управлять с помощью ручки Playback, это может быть полезно для изменения тона звука.

В данном программном комплексе предусмотрена возможность гибридной генерации, где нижняя часть спектра берется из сгенерированного звука, а верхняя часть спектра из звука входящего в датасет, частота перехода между ними регулируется ручкой Crossfade.

3.2 Подготовка авторского датасета

Для обучения данной модели использовался авторский датасет барабанных звуков, собранный за годы профильной музыкальной деятельности. Для понимания специфики датасета, в данной главе будут приведены основные сведения о барабанной установке и используемых категориях звуков.

Большинство барабанных установок состоит из нескольких отдельных барабанов, каждый из которых имеет свой характерный тембр и частотный диапазон и играет свой ритмический рисунок. Используя разные громкости и длительности для разных барабанов достигается разборчивость и отсутствие частотных конфликтов среди инструментов.



Рис. 20: Устройство барабанной установки [26].

Определенные виды барабанов, такие как томы, кики или перкуссия могут быть настроены на какой-либо частоте, другие могут не иметь конкретной настройки. Все рассматриваемые барабаны имеют разную амплитудно-частотную характеристику, их доминирующие частоты лежат в разных областях.

Kick (бас барабан) - главный барабан, который формирует основной ритм композиции, влияние которого расположено в низкочастотной части спектра. Кик состоит из двух составных частей: атаки и тела (sustain). Короткая и интенсивная часть звука, называется атакой, частотно она обычно располагается во всей части спектра. Тело бас барабана следует за атакой и обычно располагается в низкой части спектра (часто можно увидеть несущую гармонику в конце звука) и является более продолжительным, чем фаза атаки. В данной категории насчитывается 2480 разных звуков.

Snare (малый или рабочий барабан) – второй по важности барабан, ударный музыкальный инструмент, принадлежащий к мембрanoфонам с неопределенной высотой звучания. Частотно доминирует в средний части спектра. На волновой форме данного звука обычно можно обнаружить стохастическую компоненту (присутствие небольшого количества шума) и умеренное количество реверберации. Данная категория включает в себя 2986 звуков.

Closed Hi-Hat (закрытый Хай-Хэт) – короткий и высокочастотный звук играющий обычно восьмыми и шестнадцатыми долями. Обычно в звуке отсутствует низкочастотная и доминирует высокочастотная составляющая. Редко присутствуют звуки с небольшим количеством реверберации. Датасет включает в себя 1746 звуков данной категории.

Clap (хлопок) – звук, который получается при ударе в ладоши. Доминирует в верней середине спектра. Не используется в акустических барабанах, однако присутствует в драм-машинах, что объясняет его популярность в жанрах электронной музыки. В датасете находится 1261 звук из этой категории.

Приведенные ниже типы звуков возникли из реальных инструментов, однако в 80-х годах прошлого века появились драм-машины, которые алгоритмически генерируют звуки этих категорий. Таким образом датасет состоит как из записанных, так и алгоритмически сгенерированных драм-машинами звуков, часть из которых смешаны между собой с использованием эффектов постпроцессинга.

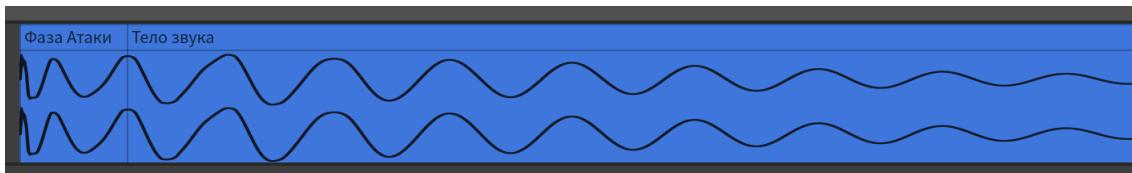


Рис. 21: Пример волновой формы бас барабана.

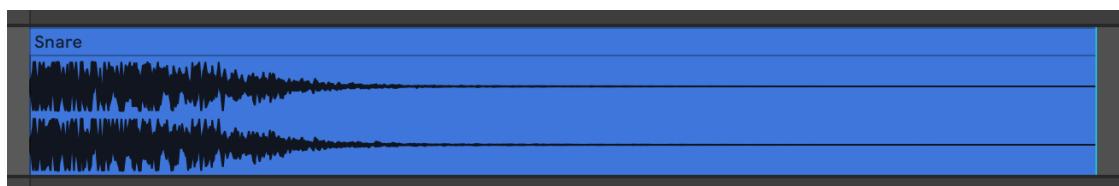


Рис. 22: Пример волновой формы малого барабана.

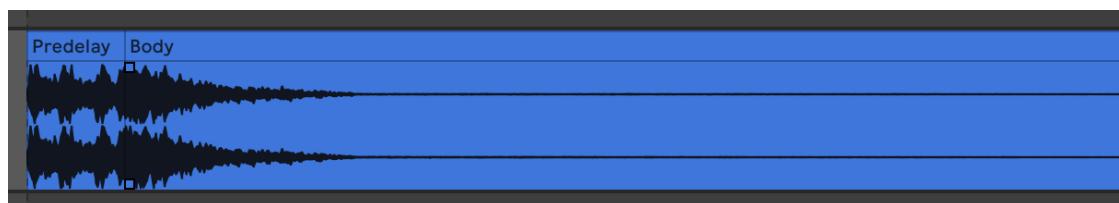


Рис. 23: Пример волновой формы хлопка

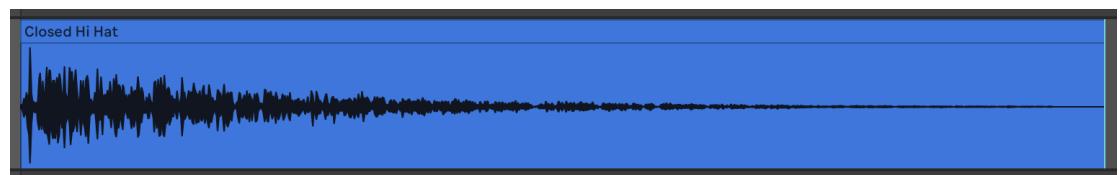


Рис. 24: Пример волновой формы закрытой тарелки.

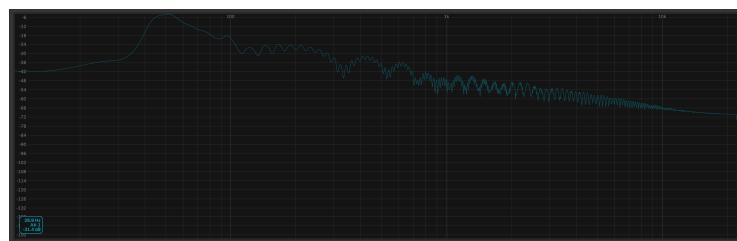


Рис. 25: Пример спектра бас барабана.

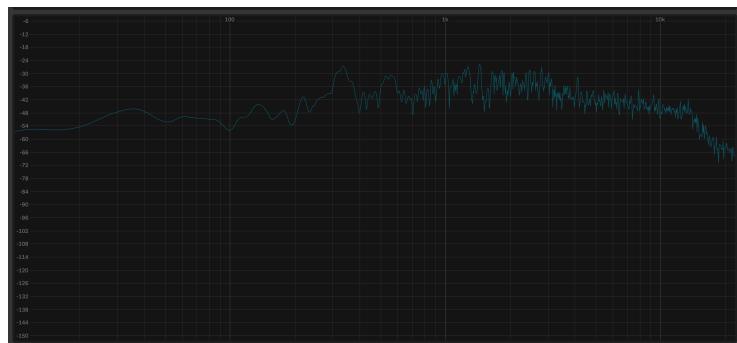


Рис. 26: Пример спектра малого барабана.

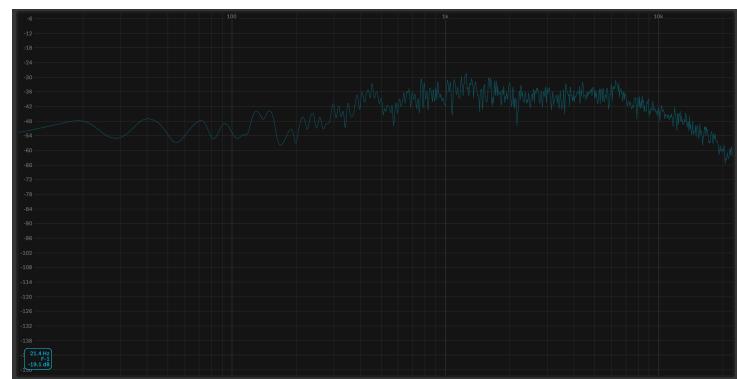


Рис. 27: Пример спектра хлопка.

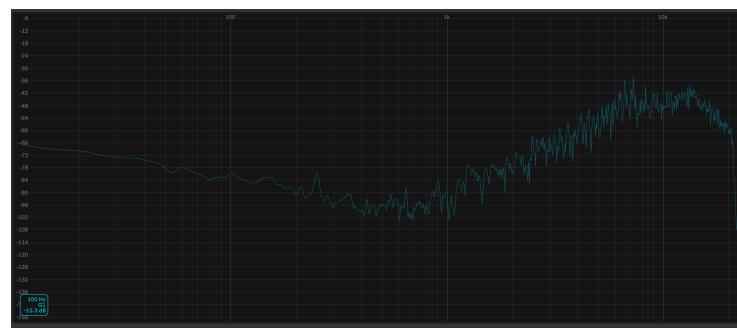


Рис. 28: Пример спектра закрытой тарелки.

3.3 Выбор и обучение нейронных сетей

При выборе нейронной сети было важно учесть несколько следующих критериев. Во-первых, данная сеть должна обладать возможностью генерации звуков в режиме реального времени (real-time), или хотя бы генерировать звуки в течении такого времени, ожидание которого не усложняло бы пользовательский опыт. В связи с этим сеть WaveNet не подходит в качестве сети для real-time генерации, так как каждый новый, сгенерированный ей сэмпл, зависит от всех предыдущих сэмплов в генерируемом объекте, таким образом на генерацию каждого звука требуется большое количество времени.

Во-вторых сама архитектура нейронной сети должна подходить под задачу генерации звуков выбранной категории. Сеть DDSP плохо подходит для моделирования звуков рассматриваемого датасета, так как барабаны из датасета чаще всего не имеют четких гармоник, и в целом, гармоническое осциллирование больше подходит для моделирования мелодических или гармонических инструментов.

В связи с этим было принято решение выбрать сеть WaveGAN и SpecGAN в качестве моделей для эксперимента. Сеть WaveGAN уже на первых тестах показывала результаты лучше, чем SpecGAN, поэтому в целях экономии вычислительных ресурсов 4 категории датасета было обучено на WaveGAN (для использования в программном комплексе) и только одна на SpecGAN для сравнения качества генерации между сетями.

Обычно сети обучаются на частоте 16 кГц, так как на это требуется существенно меньшие вычислительные ресурсы, в связи с чем теряется часть спектра и ухудшается возможное качество аудио. Однако в данной работе генерация звуков происходила на частоте 44.1 кГц в секунду, что является студийным стандартом.

Обучение моделей проводилось в облачной среде Google Colab с подключением диска Google Drive для сохранения чекпоинтов для генерации или возобновления обучения. Данный сервис очень отрицательно относится к не интерактивной работе и прерывает выполнение кода в течении 3-8 часов, если не происходит какой-то активности. В случае, если на аккаунте

часто происходит обучение сетей, после каждого отключения среды выполнения из-за неактивности, доступ к вычислительным ресурсам блокируется примерно на сутки.

В условиях ограниченных вычислительных ресурсов обучения каждой сети велось разное количество итераций, которое определялось удвоением качеством звуков на выходе сети. Таким образом для звука бас-барабана выполнилось 50000 итераций, для малого барабана 6800 итераций, для закрытой тарелки 5750 итераций, и для звука хлопка выполнилось 10500 тысяч итераций. Вполне возможно, что условия наличия больших вычислительных мощностей качество генерации звуков некоторых категорий может возрасти.

Для того чтобы размерность коротких звуков совпадала с размером входных данных добавлена команда `-data_pad_end`, которая заполняет недостающие места нулями. На случай, когда какой-либо звук превышает размерность входных данных используется команда `-data_first_slice`, которая берет только первую часть звука, которая совпадает со входным слоем.

Table 6: WaveGAN hyperparameters

Name	Value
Input data type	16-bit PCM (requantized to 32-bit float)
Model data type	32-bit floating point
Num channels (c)	1
Batch size (b)	64
Model dimensionality (d)	64
Phase shuffle (WaveGAN)	2
Phase shuffle (SpecGAN)	0
Loss	WGAN-GP (Gulrajani et al., 2017)
WGAN-GP λ	10
D updates per G update	5
Optimizer	Adam ($\alpha = 1e-4$, $\beta_1 = 0.5$, $\beta_2 = 0.9$)

Рис. 29: Другие гиперпараметры сетей WaveGAN и SpecGAN [3].

Table 2: WaveGAN generator architecture

Operation	Kernel Size	Output Shape
Input $\mathbf{z} \sim \text{Uniform}(-1, 1)$		(n , 100)
Dense 1	(100, $256d$)	(n , $256d$)
Reshape		(n , 16, 16d)
ReLU		(n , 16, 16d)
Trans Conv1D (Stride=4)	(25, 16d, 8d)	(n , 64, 8d)
ReLU		(n , 64, 8d)
Trans Conv1D (Stride=4)	(25, 8d, 4d)	(n , 256, 4d)
ReLU		(n , 256, 4d)
Trans Conv1D (Stride=4)	(25, 4d, 2d)	(n , 1024, 2d)
ReLU		(n , 1024, 2d)
Trans Conv1D (Stride=4)	(25, 2d, d)	(n , 4096, d)
ReLU		(n , 4096, d)
Trans Conv1D (Stride=4)	(25, d, c)	(n , 16384, c)
Tanh		(n , 16384, c)

Table 3: WaveGAN discriminator architecture

Operation	Kernel Size	Output Shape
Input \mathbf{x} or $G(\mathbf{z})$		(n , 16384, c)
Conv1D (Stride=4)	(25, c, d)	(n , 4096, d)
LReLU ($\alpha = 0.2$)		(n , 4096, d)
Phase Shuffle ($n = 2$)		(n , 4096, d)
Conv1D (Stride=4)	(25, d, 2d)	(n , 1024, 2d)
LReLU ($\alpha = 0.2$)		(n , 1024, 2d)
Phase Shuffle ($n = 2$)		(n , 1024, 2d)
Conv1D (Stride=4)	(25, 2d, 4d)	(n , 256, 4d)
LReLU ($\alpha = 0.2$)		(n , 256, 4d)
Phase Shuffle ($n = 2$)		(n , 256, 4d)
Conv1D (Stride=4)	(25, 4d, 8d)	(n , 64, 8d)
LReLU ($\alpha = 0.2$)		(n , 64, 8d)
Phase Shuffle ($n = 2$)		(n , 64, 8d)
Conv1D (Stride=4)	(25, 8d, 16d)	(n , 16, 16d)
LReLU ($\alpha = 0.2$)		(n , 16, 16d)
Reshape		(n , 256d)
Dense	(256d, 1)	(n , 1)

Рис. 30: Архитектура сети WaveGAN [3].

Table 4: SpecGAN generator architecture

Operation	Kernel Size	Output Shape
Input $z \sim \text{Uniform}(-1, 1)$		(n , 100)
Dense 1	(100, $256d$)	(n , $256d$)
Reshape		(n , 4, 4, $16d$)
ReLU		(n , 4, 4, $16d$)
Trans Conv2D (Stride=2)	(5, 5, $16d$, $8d$)	(n , 8, 8, $8d$)
ReLU		(n , 8, 8, $8d$)
Trans Conv2D (Stride=2)	(5, 5, $8d$, $4d$)	(n , 16, 16, $4d$)
ReLU		(n , 16, 16, $4d$)
Trans Conv2D (Stride=2)	(5, 5, $4d$, $2d$)	(n , 32, 32, $2d$)
ReLU		(n , 32, 32, $2d$)
Trans Conv2D (Stride=2)	(5, 5, $2d$, d)	(n , 64, 64, d)
ReLU		(n , 64, 64, d)
Trans Conv2D (Stride=2)	(5, 5, d , c)	(n , 128, 128, c)
Tanh		(n , 128, 128, c)

Table 5: SpecGAN discriminator architecture

Operation	Kernel Size	Output Shape
Input x or $G(z)$		(n , 128, 128, c)
Conv2D (Stride=2)	(5, 5, c , d)	(n , 64, 64, d)
LReLU ($\alpha = 0.2$)		(n , 64, 64, d)
Conv2D (Stride=2)	(5, 5, d , $2d$)	(n , 32, 32, $2d$)
LReLU ($\alpha = 0.2$)		(n , 32, 32, $2d$)
Conv2D (Stride=2)	(5, 5, $2d$, $4d$)	(n , 16, 16, $4d$)
LReLU ($\alpha = 0.2$)		(n , 16, 16, $4d$)
Conv2D (Stride=2)	(5, 5, $4d$, $8d$)	(n , 8, 8, $8d$)
LReLU ($\alpha = 0.2$)		(n , 8, 8, $8d$)
Conv2D (Stride=2)	(5, 5, $8d$, $16d$)	(n , 4, 4, $16d$)
LReLU ($\alpha = 0.2$)		(n , 4, 4, $16d$)
Reshape		(n , 256d)
Dense	(256d, 1)	(n , 1)

Рис. 31: Архитектура сети SpecGAN [3].

3.4 Анализ экспериментов

Первичный анализ результатов эксперимента заключался в измерении громкости и исследовании спектра звуков на какие-либо выбросы на графике. Так как большинство звуков из датасета откалибровано в 0 db по пику (максимально громко без искажений), то же самое ожидалось и от сгенерированных звуков. Данное предложение оправдалось и сгенерированные звуки в среднем показывают такие же значения громкости, как и образцы из датасета. Каких-либо явных аномалий и выбросов в спектре сгенерированных звуков не было обнаружено.

Анализ качества генерации аудио сложно провести с помощью каких-либо дополнительных метрик, поскольку метрики приведенные в статьях из обзора не являются актуальными применимо к данной задаче. В связи с этим анализ результатов производился с помощью экспертной оценки.

Для вынесения оценки было опрошено 5 экспертов из музыкальной индустрии, предметные области которых включают: музыкальное продюсирование, саунд-дизайн и звукорежиссуру. Оценка складывалась из двух критериев: правильная передача тембра и отсутствие критичных для студийного использования искажений. Каждому эксперту предоставлялось 50 звуков звуков для оценки (40 звуков сети WaveGAN по 10 для каждой категории и 10 звуков сети SpecGAN), после чего эксперт ставит 1, если звук удовлетворяет критериям, или ставит 0, если он не удовлетворяет критериям, после этого оценки экспертов суммируются и делятся на их количество. Такой процесс происходит для каждого звука по отдельности и далее метрики усредняются по каждой категории.

Категория звука	Передача тембра	Качество	Итог
Хлопок	0.88	0.90	0.89
Малый барабан (WaveGAN)	0.84	0.80	0.82
Закрытая тареклка	0.80	0.78	0.79
Малый барабан (SpecGAN)	0.74	0.32	0.53
Бас-барабан	0.70	0.12	0.41

Таблица 1: Результаты экспертной оценки.

При генерации из категории бас-бочки даже с учетом большого количества итераций не удалось избавится от шума в верхней части спектра (шум начинает проявляться в спектре от 600-900 Герц), анализируя волновую форму звука, стала очевидна необходимость применения алгоритмов аппроксимации с целью ее сглаживания. Для аппроксимации волновой формы был выбран фильтр Савицкого-Голея с подбором наиболее удачно звучащих параметров [27].

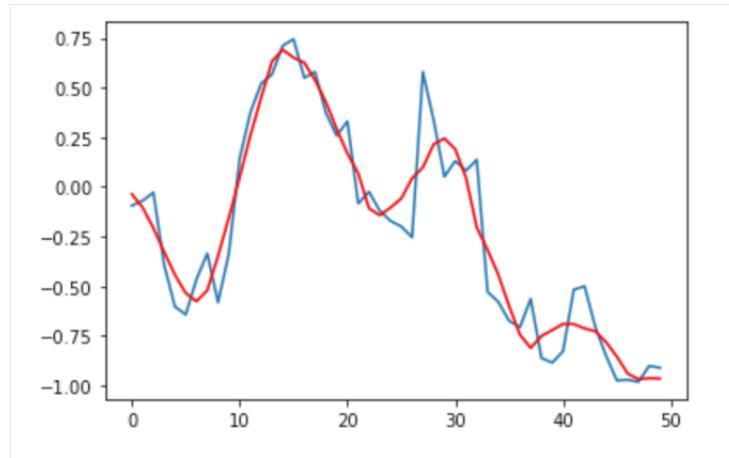


Рис. 32: Применение фильтра Савицкого-Голея к сгенерированному звуку. Синим указан изначальный звук, а красным его аппроксимация.

Некоторое количество сгенерированных звуков имеет дефекты только в конце, при том, что основная их часть звучит корректно. Для борьбы с этим эффектом можно использовать ADSR огибающую примененную к параметру громкости звука.

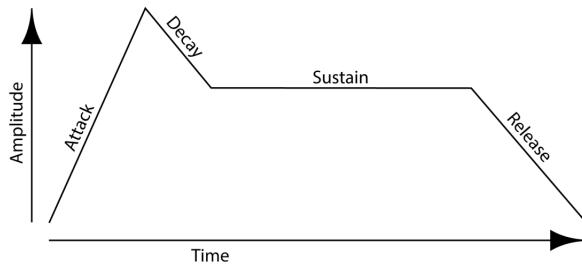


Рис. 33: Фазы ADSR огибающей [28].

Заключение

Результаты работы

В результате работы были выполнены следующие цели и задачи:

- Проведен обзор методов цифровой обработки и интеллектуального анализа аудио сигналов
- Подготовлен авторский датасет
- Произведено обучение моделей WaveGAN и SpecGAN, проведен анализ сгенерированных звуков, исследованы возможности постпроцессинга
- Разработан программный комплекс для генерации звуков в цифровой рабочей станции (Digital Audio Workstation, DAW)

Перспективы развития

Разработанный прототип имеет перспективы на то, чтобы стать коммерческим программным продуктом. Для того, чтобы его развить требуются вычислительные мощности для дообучения существующих и обучения новых сетей для категорий: перкусий, открытых тарелок, крэшей и райдов. Также можно исследовать возможность создания метрик оценки звуков и их морфинга между близкими значениями скрытых слоев, что даст больший простор для использования программного комплекса.

Также рассматривается возможность перехода от Max/MSP реализации к формату VST плагина, что позволит интегрировать данный программный комплекс в большое количество цифровых рабочих станций.

С программным кодом, сгенерированными звуками и видеообзором приложения можно ознакомиться по ссылке: <https://github.com/4antii/NGen>

Список литературы

- [1] Aaron van den Oord, Sander Dieleman and Heiga Zen. WaveNet: A Generative model for raw audio. Google DeepMind, 2016
- [2] Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, Adam Roberts. GANSynth: Adversarial Neural Audio Synthesis. ICLR 2019 Conference Blind Submission.
- [3] Chris Donahue, Julian McAuley, Miller Puckette. ADVERSARIAL AUDIO SYNTHESIS. Published as a conference paper at ICLR 2019.
- [4] Magenta: <https://github.com/magenta>
- [5] Jesse Engel, Lamtharn Hantrakul, Chenjie Gu, Adam Roberts. DDSP: Differentiable Digital Signal Processing. Published as a conference paper at ICLR 2020.
- [6] Mubert: <https://mubert.com/>
- [7] Soniebe Smart EQ: <https://www.sonible.com/smarteq3>
- [8] Soundtheory Gullfoss: <https://www.soundtheory.com/home>
- [9] Izotope Mixing Assistant: <https://www.izotope.com/en/products/neutron.html>
- [10] Izotope Mastering Assistant: <https://www.izotope.com/en/products/ozone.html>
- [11] LANDR: <https://www.landr.com/>
- [12] NeuralDSP Quad Cortex: <https://neuraldsp.com/quad-cortex>
- [13] Splice Sounds: <https://splice.com/features/sounds>
- [14] Аудиоформаты: https://en.wikipedia.org/wiki/Audio_file_format
- [15] Теория звука. Что нужно знать о звуке, чтобы с ним работать. Опыт Яндекс.Музыки: <https://habr.com/ru/company/yandex/blog/270765/>

- [16] Теорема Котельникова-Найквиста: https://studme.org/171390/tehnika-teorema_kotelnikova_teorema_otschetov
- [17] Спектроанализаторы: <https://prosound.ixbt.com/education/spektr-analys.shtml>
- [18] Измерение громкости: <https://www.newtoneacademy.com/rmsvslufs>
- [19] ПО для анализа громкости: <https://www.izotope.com/en/products/insight.html>
- [20] Мю-закон: <http://digitalsoundandmusic.com/5-3-8-algorithms-for-audio-companding-and-compression/>
- [21] Goodfellow, Ian J.; Pouget-Abadie, Jean; Mirza, Mehdi; Xu, Bing; Warde-Farley, David; Ozair, Sherjil; Courville, Aaron Bengio, Yoshua (2014), Generative Adversarial Networks. Universite de Montreal.
- [22] Martin Arjovsky, Soumith Chintala, Léon Bottou. Wasserstein GAN. Facebook AI Research. 2017.
- [23] Шахматный эффект: <https://distill.pub/2016/deconv-checkerboard/>
- [24] Архитектура DCGAN: https://gluon.mxnet.io/chapter14_generative-adversarial-networks/dcgan.html
- [25] Клиент-сервер Архитектура: <https://habr.com/ru/post/495698/>
- [26] Компоненты барабанной установки: https://www.muzbar.ru/articles/Kak_vybrat_barabannuyu_ustanovku/?page=7
- [27] Фильтр Савицкого-Голея: <https://www.mathworks.com/help/signal/ref/sgolayfilt.html>
- [28] ADSR огибающая: <https://jythonmusic.me/envelope/>