ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет информатики, математики и компьютерных наук

**Программа подготовки магистров по направлению 01.04.02
Прикладная математика и информатика**

Магистерская программа «Интеллектуальный анализ данных»

Ляхов Даниил Андреевич

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Клонирование голоса

Рецензент:
Старший преподаватель
Дурандин Олег Владимирович

Научный руководитель:
Старший преподаватель
Досов Санжар Музаффарович

Нижний Новгород, 2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Клонирование голоса – популярная задача, используемая для

- помощи людям, потерявщим свой голос;
- автоматическое озвучивания фильмов / аудиокниг;
- реалистичного перевода с сохранением голоса говорящего.

Известный метод решения данной задачи - нейронные сети, решающие сразу несколько подзадач внутри себя: генерация голоса (Text To Speech), идентификация спикера (Speaker Identification) и вокализации MEL-спектрограмм. Объединяя эти три задачи и подготавливая специальный датасет, задача в некотором приближении была решена на английском языке[1]. Задача данной работы - реализация алгоритма клонирования голоса на русском языке в условиях ограниченности обучающего набора данных. Код работы в открытом доступе доступен на github[29].

1 Обработка звуковых сигналов

1.1 Цифровое представление звуковых сигналов

Звук — это вибрация молекул воздуха или изменение давления воздуха, которое можно уловить ухом [2]. Характер и частота слышимых вибраций придают звуку его уникальное качество. Диапазон слухового восприятия составляет примерно от 20 циклов (или колебаний) давления воздуха в секунду до 20000 циклов в секунду. Колебания давления воздуха вне этого диапазона не слышны человеческому уху и называются дозвуковыми (менее 20 циклов в секунду) и ультразвуковыми (более 20000 циклов в секунду) колебаниями. Звук создается благодаря силе трения, например во время удара барабанной палочки по тарелке, движения смычка по струне скрипки, или вибрация диффузора динамика, который приводит в движение окружающие молекулы воздуха. От точки удара или возмущения звуковые волны или узоры вибрирующих молекул воздуха излучаются наружу через атмосферу к уху, как рябь воды в пруду.

Когда звуковые волны, издаваемые виолончелью, быстро распространяются через атмосферу, они отражаются от различных поверхностей, разделяются и умножаются на множество отражений. Как только эти отражения достигают уха, они преобразуются в электрические нервные импульсы и отправляются в мозг, где они хранятся и интерпретируются как звук «виолончели». Точно так же эти колебания или отражения давления воздуха могут быть преобразованы в электрические волны или сигналы с помощью микрофона и отправлены на записывающее устройство, которое хранит образец «формы волны». Уникальная картина колебаний давления воздуха или звуковых отражений,

создаваемых инструментом или диффузором динамика, известна как формы волны (wave form).

Громкость звука — это восприятие силы или слабости звуковой волны в зависимости от степени производимого давления. Звуковые волны с большей интенсивностью или большими перепадами давления воздуха производят более громкие звуки. Звуковые волны с меньшими колебаниями атмосферного давления производят более тихие звуки. Интенсивность волны влияет на сочетание звуков. Чем громче звук, тем больше он будет маскировать или доминировать над другими звуками в общем звуке.

Высота тона — это психоакустический термин, обозначающий, насколько высокий или низкий звук воспринимается человеческим ухом. Высота звука определяется частотой звука. Средняя «до» на фортепиано, например, вибрирует с частотой 261 цикл в секунду. Частота измеряется в герцах или Гц = циклов в секунду. Чем выше частота, тем выше тон. Чем ниже частота, тем ниже тон. Для воспроизведения звука высокого тона требуется больше циклов за тот же период времени, чем для воспроизведения звука низкого тона. Таким образом, для точного воспроизведения высоких звуков, таких как женский голос или жужжание мухи, требуется больше цифровой информации, чем для более низких звуков, таких как мужской голос или бас-гитара. По этой причине низкие звуки меньше портятся в процессе преобразования звука (кодирования) в формат низкого качества.

Большинство звуков представляют собой смесь волн различных частот. Например, нота виолончели состоит из множества частот в частотном спектре. Спектр частот — это полный диапазон частот, которые может уловить человеческое ухо, точно так же, как цветовой спектр — это

диапазон цветов, которые может уловить человеческий глаз. Этот термин также используется по отношению к конкретному звуку, означая, что частотный спектр представляет собой диапазон частот, присутствующих в этом звуке. Например, средняя нота виолончели имеет диапазон от 500 Гц до 12000 Гц.

Каждая нота состоит из набора «высот», которые вибрируют в гармонии с ее основной частотой или высотой тона. Музыкальные тона содержат несколько таких звуков, известных как гармоники. Это явление можно испытать как на слух, так и визуально, например слушая и наблюдая за перебором гитарной струны. Струна будет вибрировать не только на основной частоте, но и на более высоких кратных этой частоте. Эти дополнительные частоты являются гармониками. Например, нота До, исполняемая в средней тональности на виолончели, будет преимущественно резонировать с частотой 261 цикл в секунду, но также будет содержать частоты с частотой 1000, 2000 и 4000 циклов в секунду.

Преобразования звука в цифровую информацию производится при помощи микрофона - устройства, способного преобразовывать давление воздуха на мембрану в изменение напряжения в электрической сети своего устройства, которые возможно представить в дискретном виде и передать для обработки далее как цифровой сигнал. Две основные характеристики при оцифровывании звуковых волн - частота и глубина дискретизации. Количество замеров микрофона, сделанных в секунду, называется частотой дискретизации и измеряется в Герцах. Глубина дискретизации есть количество бит, в которое полученные значения кодируются, измеряется в битах. Пример оцифровки сигнала изображен на рисунке 1.1.

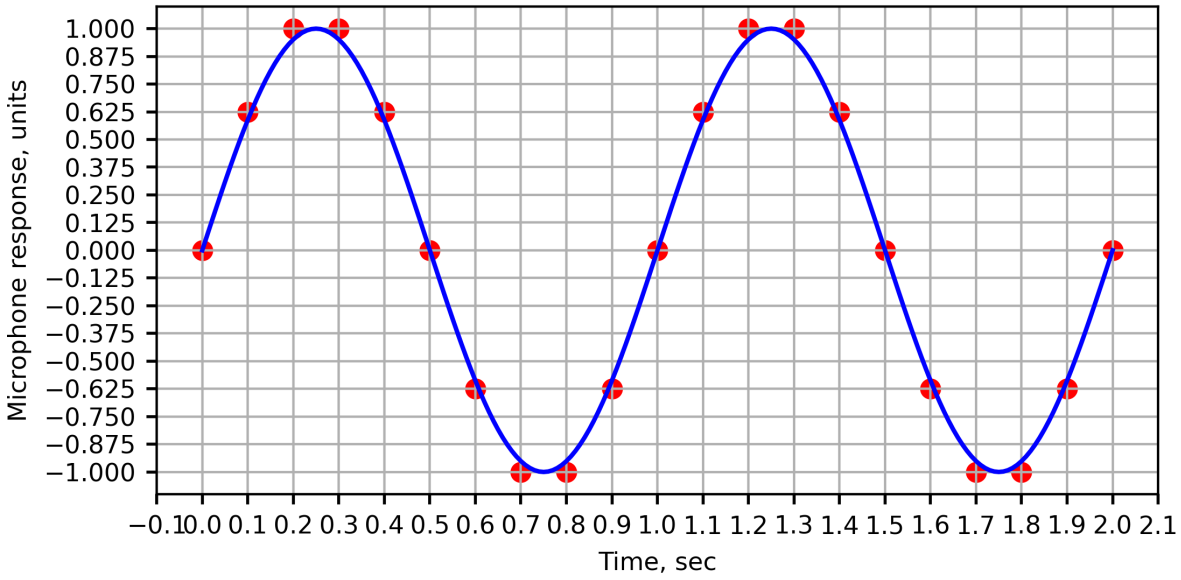


Рисунок 1.1 – Оцифровка синусоиды с частотой дискредитации 10Гц и глубиной 4 бита. Синим изображен исходный сигнал, красные точки - полученные в результате дискредитации значения.

Для извлечения входящих в звуковой сигнал базовых частот используется оконное дискретное преобразование Фурье, описанное в главе 1.2.1. Для визуализации базовых частот широко используются спектрограммы и MEL спектрограммы, описанные в главе 1.2.2.

1.2 Методы анализа звукового сигнала

1.2.1 Преобразование Фурье

Непрерывное преобразование Фурье имеет вид:

Прямое:

$$F(s) \equiv \mathcal{F}\{f(x)\}(s) \equiv \int_{-\infty}^{\infty} f(x)\, e^{-2\pi i s x}\, dx \tag{1.1}$$

Обратное:

$$f(x) \equiv \mathcal{F}^{-1}\{F(s)\}(x) \equiv \int\limits_{-\infty}^{\infty} F(s)\, e^{2\pi i s x}\, ds \tag{1.2}$$

Где $f(x)$ - непрерывная функция - сигнал вещественной переменной, $F(x)$ - непрерывная функция комплексной переменной - Фурье образ $f(x)$ [3].

Сверткой называют интегральное преобразование вида:

$$(f * g)(x) = \int\limits_{-\infty}^{\infty} f(y)\, g(x-y)\, dy \; = \int\limits_{-\infty}^{\infty} f(x-y)\, g(y)\, dy \tag{1.3}$$

Корреляцией двух функций называют преобразование вида:

$$(f \star g)(x) = \int\limits_{-\infty}^{\infty} f(y)\, g(x+y)\, dy \; = \int\limits_{-\infty}^{\infty} f(x+y)\, g(y)\, dy \tag{1.4}$$

Для непрерывного преобразования Фурье доказана теорема свертки

$$(f * g)(x) \equiv \mathcal{F}^{-1}\{\mathcal{F}(f) \cdot \mathcal{F}(g)\} \tag{1.5}$$

Аналогично теореме о свертке, существует теорема о корреляция

$$(f \star g)(x) \equiv \mathcal{F}^{-1}\{\mathcal{F}^{*}(f) \cdot \mathcal{F}(g)\} \tag{1.6}$$

Где операция $\cdot$ – скалярное произведение, а $\mathcal{F}^{*}(f)$ - комплексно сопряженный образ Фурье функции f .

Представим звуковой сигнал как функцию $f(x)$, принимающую ненулевые значения в промежутке $(0, T)$, значения $f(0) = \frac{\lim_{x \to +0} f(x)}{2}$ $f(T) = \frac{\lim_{x \to -T} f(x)}{2}$. Не каждая функция может быть представлена в частотном базисе путем преобразования Фурье, для этого должен быть выполнен ряд условий[3]. Поэтому также предположим, что $f(t)$ имеет непрерывный Фуьре образ $F(\mu)$, принимающий ненулевые значения только на отрезке $[-\mu_{max}, \mu_{max}]$.

Как было описано в главе 1.1, во время записи звукового сигнала возможно сохранение только конченого числа точек во времени. Выбирая частоту дискретизации ΔT, можно представить набор сэмплированных точек в виде:

$$\tilde{f}(t) = f(t)s_{\Delta T}(t) = \sum_{n=-\infty}^{\infty} f(t)\delta(t - n\Delta T) \tag{1.7}$$

Где δ есть функция Дирака. Дискретная функция $\tilde{f}(t)$ имеет Фурье образ $\tilde{F}(\mu)$

$$\tilde{F}(\mu) = \frac{1}{\Delta T} \sum_{n=-\infty}^{\infty} F(\mu - \frac{n}{\Delta T}) \tag{1.8}$$

Из формулы видно, что $\tilde{F}(\mu)$ является непрерывной и периодической функцией с периодом $\frac{1}{\Delta T}$. Из предположения о том, что $F(\mu)$ принимает ненулевые значения только на отрезке $[-\mu_{max}, \mu_{max}]$ и выбирая $\Delta T = \frac{1}{2\mu_{max}}$, можем выразить $F(\mu)$ из $\tilde{F}(\mu)$ формулой:

$$F(\mu) = H(\mu)\tilde{F}(\mu) \tag{1.9}$$

Где

$$H(\mu) = \begin{cases} \Delta T & \mu \in [-\mu_{max}, \mu_{max}] \\ 0 \end{cases} \tag{1.10}$$

Обратное преобразование Фурье для $H(\mu)$ имеет вид

$$h(t) = \mathcal{F}^{-1}\{H(\mu)\}(t) = \int_{-\infty}^{\infty} H(\mu) e^{2\pi i \mu t} d\mu = \int_{-\mu_{max}}^{\mu_{max}} \Delta T e^{2\pi i \mu t} d\mu =$$

$$= \frac{\Delta T}{i 2\pi t} \left[e^{2\pi i \mu t} \right]_{-\mu_{max}}^{\mu_{max}} = 2\Delta T \mu_{max} \frac{sin(2\pi t \mu_{max})}{(2\pi t \mu_{max})} = 2\Delta T \mu_{max} sinc(2t\mu_{max})$$

Тогда исходная функция $f(t)$ может быть получена из следующего соотношения:

$$f(t) = \mathcal{F}^{-1}\{F(\mu)\}$$
$$= \mathcal{F}^{-1}\{H(\mu)\tilde{F}(\mu)\}$$
$$= h(t) * \tilde{f}(t)$$

где последнее неравенство получено благодаря теореме свертки для пар образов Фурье. Соотношение можно переписать как

$$f(t) = \sum_{n=-\infty}^{\infty} f(n\Delta T) sinc[(t - n\Delta T)/\Delta T] = \sum_{n=0}^{\frac{T}{\Delta T}} f(n\Delta T) sinc[(t - n\Delta T)/\Delta T] \tag{1.11}$$

Таким образом, выбирая частоту дискретизации $\Delta T = \frac{1}{2\mu_{max}}$, для сохранения непрерывной функции $f(t)$, удовлетворяющей всем заданным

условиям, достаточно хранить $\frac{T}{\Delta T} + 1$ значений этой функции. Данный вывод называется теоремой Котельникова или сэмплирования.

$\tilde{F}(\mu)$ есть непрерывная ограниченная функция

$$\tilde{F}(\mu) = \int_{-\infty}^{\infty} \tilde{f}(t) e^{-j2\pi\mu t} dt \tag{1.12}$$

$$\begin{aligned} \tilde{F}(\mu) &= \int_{-\infty}^{\infty} \tilde{f}(t) e^{-j2\pi\mu t} dt = \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f(t)\delta(t - n\Delta T) e^{-j2\pi\mu t} dt \\ &= \sum_{n=-\infty}^{\infty} \int_{-\infty}^{\infty} f(t)\delta(t - n\Delta T) e^{-j2\pi\mu t} dt \\ &= \sum_{n=-\infty}^{\infty} f_n e^{-j2\pi\mu n\Delta T} = \sum_{n=0}^{\frac{T}{\Delta T}} f_n e^{-j2\pi\mu n\Delta T} \end{aligned} \tag{1.13}$$

Для того, чтобы описать $\tilde{F}(\mu)$ достаточно сохранить всего $M = \frac{T}{\Delta T} + 1$ значений функции. Чтобы показать это, рассмотрим равномерную сетку частот

$$\mu = \frac{m}{M\Delta T} \quad m = 0, 1, 2, \ldots, M - 1 \tag{1.14}$$

Принимая во внимание тот факт, что $\tilde{F}(\mu)$ периодическая с периодом ΔT, что видно из формулы 1.8, достаточно сохранить значения только для одного из периодов этой функции. Тогда, подставляя 1.14 в формулу 1.13, получаем прямое дискретное преобразование Фурье:

$$F_m = \sum_{n=0}^{M-1} f_n e^{-j2\pi mn/M} \quad m = 0, 1, 2, \ldots, M - 1 \tag{1.15}$$

Обратное дискретное преобразование Фурье имеет вид

$$f_n = \frac{1}{M}\sum_{m=0}^{M-1} F_m e^{j2\pi mn/M} \quad n = 0, 1, 2, \ldots, M-1 \tag{1.16}$$

Можно показать, что при подстановке 1.15 в 1.16 можно получить равенство $f_n = f_n$, аналогично, подставляя 1.16 в 1.15 можно получить равенство $F_m = F_m$. Таким образом последовательности $\{f_m\}$ взамиооднозначно ставиться в соответствие последовательность образов $\{F_m\}$. Если частота дискретизации исходного сигнала равна ΔT, то шаг сетки частот будет равен $\Delta\mu = \frac{1}{M\Delta T} = \frac{1}{T+\Delta T}$

Для случая дискретного преобразования Фурье, также как и для непрерывного случая, действует теорема свертки[4], позволяющая улучшить асимптотику операций свертки и кросс корреляции используя для этого переход в частотный базис.

Последовательность $\{F_m\}$ есть комплексные коэффициенты дискретного ряда Фурье. Переходя в полярные координаты $F_m = F_m^r + j \cdot F_m^j = \mid F_m \mid \cdot e^{j\theta_m}$, можно представить прямое дискретное преобразование Фурье следующим образом:

$$\begin{aligned} f_n = \frac{1}{M}\sum_{m=0}^{M-1} F_m e^{j2\pi mn/M} &= \frac{1}{M}\sum_{m=0}^{M-1} \mid F_m \mid \cdot e^{j\theta_m} \cdot e^{j2\pi mn/M} \\ &= \frac{1}{M}\sum_{m=0}^{M-1} \mid F_m \mid \cdot e^{j(2\pi mn/M + \theta_m)} \end{aligned} \tag{1.17}$$

$$e^{j(\omega\cdot n+\theta_m)} = \sin(\omega \cdot n + \theta_m) + j\cos(\omega \cdot n + \theta_m)$$

Из равенства видно, что $\mid F_m \mid$ отвечают за магнитуду, входимую в исходный сигнал соответствующей базовой частоты, и θ_m отвечает за сдвиг этой базовой частоты или фазу.

На практике для частотного анализа сигналов их разбивают на отдельные части - окна, для каждого из которых находятся коэффициенты ряда Фурье. Последовательность полученных коэффициентов используется как приближение разложения в ряд Фурье всей последовательности.

Оконное дискретное преобразование Фурье (Short Time Discrete Fourier Transformation)[4] определяется как

$$\mathrm{STFT}\{f(n)\}(m_k, \omega) \equiv F(m_k, \omega) = \sum_{n=0}^{M-1} f(n) w(n - m_k) e^{-j\omega n} \qquad (1.18)$$

где $f(n)$ - исходный дискретный сигнал, $w(n)$ - оконная функция, m_k - последовательность сдвигов оконной функции.

1.2.2 Спектрограмма и её разновидности

Спектрограмма или магнитудная спектрограмма — это визуальное представление спектра частот сигнала, меняющегося во времени[5], определяется как

$$\mathrm{spectrogram}(t, \omega) = |\mathrm{STFT}(t, \omega)|^2 \qquad (1.19)$$

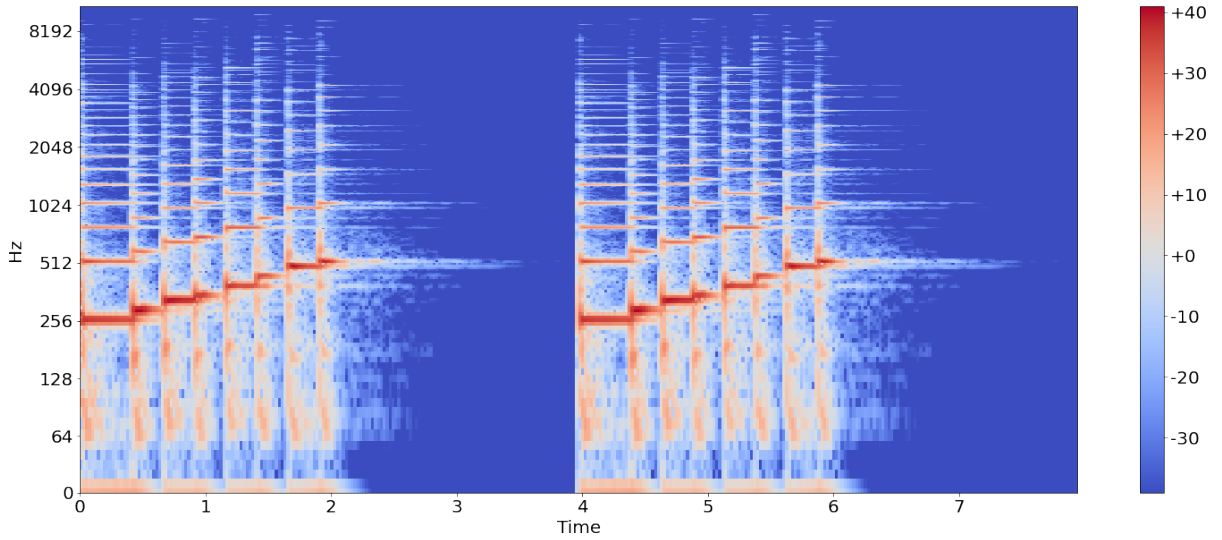


Рисунок 1.2 – Пример магнитудной спектрограммы аудиозаписи фортепиано

Как видно из формулы, спектрограмма сохраняет только магнитуды базовых частот, не учитывая влияние сдвига для каждой базовой частоты. Спектрограмму можно представить в виде матрицы $S_{M \times W}$ где M - количество частот в дискретном разложении Фурье и W - количество окон, необходимое для покрытия данного на вход файла. Такое упрощенное представление позволяет визуализировать распределение магнитуд частот исходного сигнала и упрощает его обработку. Недостаток использования такого подхода обработки сигналов - невозможность точного обратного преобразования спектрограммы в исходный сигнал.

Стандартным способом представления звуковых сигналов является MEL- спектрограмма[6]. Популярность этого метода обусловлена тем, что MEL базис отражает восприимчивость человеческого уха на частоты определенной высоты: из экспериментов обнаружено, что низкие звуковые частоты человеческий слух отличает лучше, чем высокие. Базис MEL частот выражается следующей формулой:

$$\text{Mel}(f) = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \tag{1.20}$$

Для того, чтобы посчитать MEL спектрограмму на основе магнитудной спектрограммы, необходимо сперва вычислить Filter bank - матрицу коэффициентов для фильтрации исходного частотного представления сигнала[6]. Первоначально задается параметр - количество фильтров P. В частотном базисе равномерно относительно MEL шкалы определяется сетка из точек, количество которых равно количеству фильтров. Как видно на рисунке 1.3, каждому фильтру соответствует точка в сетке, в которой значение фильтра равно 1. Проводятся две прямые: из предыдущей точки сетки в текущую и из текущей точки в следующую, при этом в соседних точках фильтр принимает значение 0.

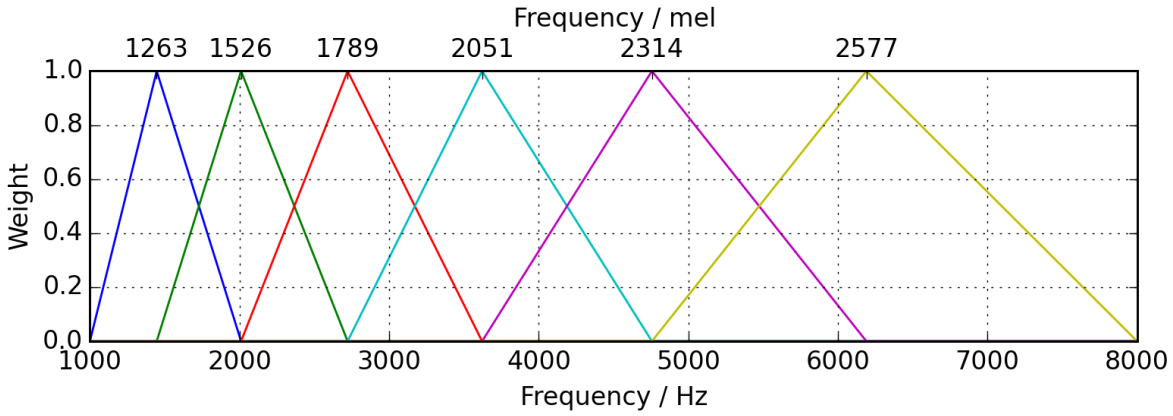


Рисунок 1.3 – Пример Filter bank для $P = 6$

Полученный фильтр - матрица коэффициентов $\text{filter}_{P \times M}$, в каждой строчке содержит соответствующий фильтр для определенной частоты, в то время как магнитудная спектрограмма $S_{M \times W}$ в каждом столбце содержит значения магнитуд базовых частот для соответствующих окон.

Фильтр применяется к спектрограмме следующим образом:

$$\text{Mel} = filter \times S \tag{1.21}$$

Результат применения такого фильтра - матрица размерности $M \times W$ - MEL коэффициенты для каждого из окон спектрограммы

Другой способ представления сигналов - мел-кепстральные коэффициенты (Mel-frequency cepstral coefficients - MFCC)[6][7][8]

$$\text{MFCC}_i = \sum_{k=0}^{P-1} \text{Mel}[k,:] \cos\left[i\left(k - \frac{1}{2}\right)\frac{\pi}{P}\right], \quad i = 1, 2, \cdots, M_{MFCC} \tag{1.22}$$

Матрица $\text{MFCC}_{M_{MFCC} \times W}$ получается из MEL спектрограммы путем применения к ней косинусного преобразования[8]

2 Клонирование голоса

2.1 Постановка задачи

Дан короткий звуковой файл длиной 5-10 секунд, необходимо преобразовать текст на русском языке в голос так, как если бы этот текст читал человек, записанный на входном звуковом сигнале. Задача дополняет проблему генерации голоса по тексту (Text To Speech, TTS) условием схожести выходного голоса с голосом, переданным на вход.

Датасеты для решение задачи клонирования голоса представляют из себя такие признаки как:

- номер (id) спикера;
- аудиозапись длинной 3-10 секунд;
- текстовая расшифровка аудиозаписи.

Автор работы [1] предлагает разбиение задачи клонирования голоса на 2 подзадачи: 1) Идентификация спикера по голосу 2) Условная генерация голоса по тексту

Для того, чтобы параметризовать задачу генерации голоса, автор предлагает использовать закодированное представление спикера из задачи идентификации, которое используется как дополнительный признак генеративной модели.

Популярный тренд развития области TTS - разбиение моделей генерации на две части: предсказание MEL-спектрограммы и предсказание непосредственно звукового сигнала на основе полученной MEL спектрограммы.

Существует известнае модификации репозитория [1], автор которого попытался решить поставленную задачу для русского языка[30], однако

результаты работы субъективно уступают качеству английской версии модели. Возможные причины субъективно низкого качества генерации - чрезмерное усложнение модели, приводящее к переобучению на небольшом тренировочном наборе и использование тренировочных данных низкого разрешения, что может привести к переобучению блока генерации речи (TTS) на имеющиеся в датасете искажения и шумы.

В рамках работы предлагается использовать оригинальную архитектуру моделей из репозитория Voice Cloning[1] и выбрать для тренировки датасеты с аудиозаписями высокого разрешения. Подробности выбора данных описаны в главе 3.1.

К сожалению, на сегодняшний день не существует объективной метрики оценки качества генерации голоса кроме только значения функции ошибки [1], поэтому для валидации работы используется субъективная метрика ”Средняя оценка эксперта (Mean Option Score)”.

2.2 Связанные задачи

2.2.1 Задача идентификация спикера

Даны набор аудиозаписей, на которых записаны голоса разных спикеров, причем в каждой аудиозаписи присутствует голос только одного спикера. Необходимо разметить аудиозаписи так, чтобы каждый элемент набора получил метку, причем записи, произведенные одним спикером, должны иметь одинаковую метку, а записи, произведенные разными спикерами - разные.

Для решения данной задачи хорошо зарекомендовал себя подход, основанный на моделях LSTM [9] и обобщенной функции ошибки для задачи верификации спикера (Generalized End to End Loss, GE2E)[10].

В качестве признаков используются MEL-спектрограммы, полученные из входных аудиозаписей. Элементы датасета имеют вид $\{\mathbf{x}_{j,i}\}$, где j есть индекс спикера, а i есть индекс фразы этого спикера. GE2E подход появился как улучшение предшествующего метода - Tuple Based End to End Loss (TE2E), суть которого заключается в следующем: входные признаки делятся на пары $\{\mathbf{x}_{j\sim}, (\mathbf{x}_{k1}, \ldots, \mathbf{x}_{kM})\}$. Первый элемент и M элементов второго элемента пары преобразовываются в латентное пространство посредством LSTM модели [9] и нормированием по L2 норме в пару вида $\{\mathbf{e}_{j\sim}, (\mathbf{e}_{k1}, \ldots, \mathbf{e}_{kM})\}$. Для того, чтобы сравнить вектор спикера j со всеми векторами спикера k, набор $(\mathbf{e}_{k1}, \ldots, \mathbf{e}_{kM})$ агрегируется посредством поэлементного усреднения:

$$\mathbf{c}_k = E_m \left[\mathbf{e}_{km}\right] = \frac{1}{M} \sum_{m=1}^{M} \mathbf{e}_{km} \tag{2.1}$$

Значение $\mathbf{c_k}$ называется центроидом спикера k. Вычисляя схожесть векторов $\mathbf{x}_{j\sim}$ и $\mathbf{c}_k$ при помощи косинусного расстояния с обучающимися парамметрами w и b

$$s = w \cdot \cos\left(\mathbf{e}_{j\sim}, \mathbf{c}_k\right) + b \tag{2.2}$$

итоговая функция ошибки для сети выглядит следующим образом:

$$L_{\mathrm{T}}\left(\mathbf{e}_{j\sim}, \mathbf{c}_k\right) = \delta(j,k)(1-\sigma(s)) + (1-\delta(j,k))\sigma(s) \tag{2.3}$$

В качестве альтернативы авторы [10] предложили сравнивать не только один признак j спикера, но сравнивать сразу N спикеров, каждый

из которых имеет M фраз, друг с другом попарно. Для этого исходный набор признаков $\{\mathbf{X}_{j,i}\}, j = 1 \cdots N, i = 1 \cdots M$ транслируется в латентное пространство при помощи LSTM модели:

$$\mathbf{e}_{ji} = \frac{f\left(\mathbf{x}_{ji}; \mathbf{w}\right)}{\left\|f\left(\mathbf{x}_{ji}; \mathbf{w}\right)\right\|_2} \tag{2.4}$$

где f - функция модели LSTM, а $\mathbf{w}$ - её веса. Затем, также с использованием обучаемых параметров w и b, вычисляется тензор схожести $\mathbf{S}_{ji,k}$:

$$\mathbf{S}_{ji,k} = \begin{cases} w \cdot \cos\left(\mathbf{e}_{ji}, \mathbf{c}_j^{(-i)}\right) + b & \text{if} \quad k = j \\ w \cdot \cos\left(\mathbf{e}_{ji}, \mathbf{c}_k\right) + b & \text{otherwise} \end{cases} \tag{2.5}$$

$$\mathbf{c}_j^{(-i)} = \frac{1}{M-1} \sum_{\substack{m=1 \\ m \neq i}}^{M} \mathbf{e}_{jm} \tag{2.6}$$

Элемент тензора ji, k отвечает за схожесть i фразы j спикера с центроидом спикера k. Для того, чтобы непосредственно посчитать функцию ошибки, авторы[10] предлагают два возможных подхода: Softmax:

$$L\left(\mathbf{e}_{ji}\right) = -\mathbf{S}_{ji,j} + \log \sum_{k=1}^{N} \exp\left(\mathbf{S}_{ji,k}\right) \tag{2.7}$$

Контрастная функция ошибки:

$$L\left(\mathbf{e}_{ji}\right) = 1 - \sigma\left(\mathbf{S}_{ji,j}\right) + \max_{\substack{1 \leq k \leq N \\ k \neq j}} \sigma\left(\mathbf{S}_{ji,k}\right) \tag{2.8}$$

Итоговое значение ошибки для всего тензора схожести определяется так:

$$L_G(\mathbf{x}; \mathbf{w}) = L_G(\mathbf{S}) = \sum_{j,i} L\left(\mathbf{e}_{ji}\right) \tag{2.9}$$

В случае функции ошибки типа Softmax, чем больше вектора, относящиеся к разным спикерам, больше похожи друг на друга, тем больше ошибка. Но так как задача, кроме всего прочего, требует, чтобы вектора одного спикера были похожи друг на друга - из суммы вычитается значение $\mathbf{S}_{ji,j}$. Так как производная нового отрицательного слагаемого, начиная с некоторого момента, больше, чем доля, вносимая $\mathbf{S}_{ji,j}$ в сумму логарифмов - можно сказать, что ошибка тем больше, чем ближе вектора разных спикеров и тем меньше, чем ближе вектора одного и того же спикера. То же самое можно сказать о контрастной версии ошибки для данной задачи.

Сравнивая GE2E и TE2E подходы, авторы приводят следующие аргументы в пользу обновленной версии: во время вычисления $L(\mathbf{e}_{ij})$ в GE2E используется $N \times M$ векторов спикеров одновременно, что приводит к тому, что во время обратного распространения ошибки путем градиентного спуска все $N \times M$ векторов стремятся в желаемом направлении: вектора одного спикера сжимаются в кластер, а разных спикеров - отдаляются. В то время как аналогичный шаг в TE2E задействует только лишь $M + 1$ вектор. Таким образом, можно ожидать, что подтверждается экспериментами, что модель сходится к лучшему результату за меньшее число шагов тренировки.

2.2.2 Задача преобразование текста в речь

Дан текст на естественном языке. Необходимо сгенерировать аудиозапись, содержащую человеческую речь, воспроизводящую данный на вход текст. Задача входит в класс проблем трансляции последовательности (текст) в последовательность (аудиозапись) (Sequence to Sequence).

Часто используемый метод решения данной задачи - модель Tacotron[11], имеющая на данный момент две версии реализации. Модель состоит из энкодера текста(encoder), блока внимания (attention block) и декодера(decoder), который генерирует непосредственно столбцы MEL спектрограммы. Схему архитектуры модели изображена на рисунке2.2

Первая подзадача - транслирование текста на естественном языке в репрезентативное латентное пространство, для чего используется блок-энкодер. В качестве входных элементов используются отдельные символы, которые при помощи операции embedding[12] преобразовываются в вещественные вектора. Далее к каждому полученному вектору поэлементно применяется нелинейная операция - dropout[12]. После этого набор векторов передается в подсеть CBHG.

Первый этап подсети CBHG - свертка входной последовательности с набором конволюций $C_k, k = 1 \cdots K$, где C_k есть одномерная конволюция с ядром размерности k. Параметр stride конволюций равен 1, поэтому на выходе этого блока для каждого символа строится K векторов, из которых затем путем применения операции max pooling для каждого входного символа выбирается наибольшее по модулю значение сверток. Таким образом, после прохождения блока наборов конволюций выход модели для каждого из элементов обладает контекстной информацией.

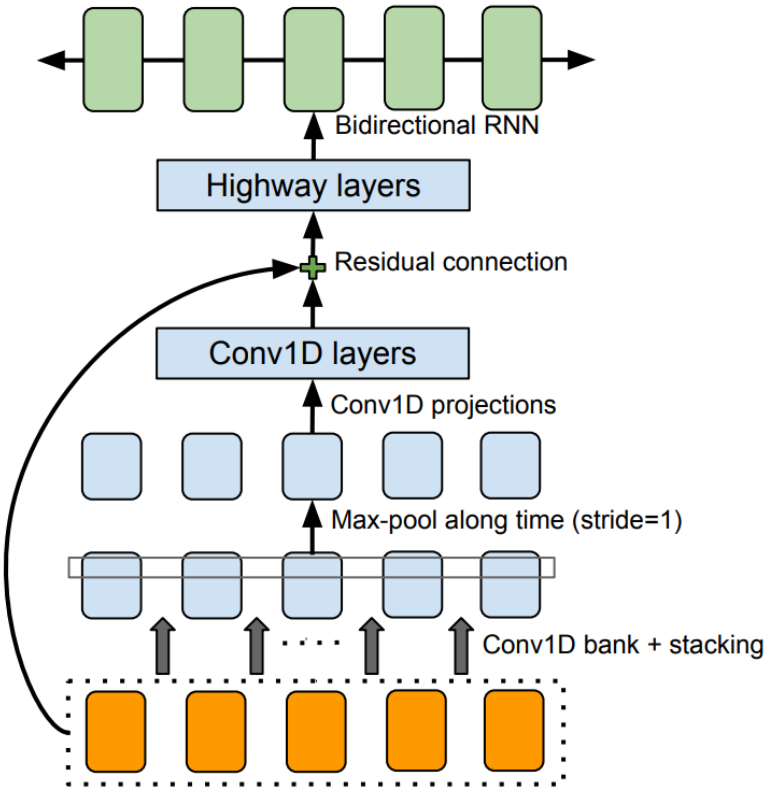


Рисунок 2.1 – Архитектура блока CBHG

После операции max pooling к последовательности применяется несколько одномерных конволюций с фиксированной размерностью фильтров, и входная последовательность складывается полученной последовательностью векторов, что называется residual connection, что позволяет улучшить сходимость модели благодаря уменьшению влияния проблему резкого роста / затухания градиента[13].

Далее к полученному вектору применяется блок, называемый highway networks [14], который можно описать следующим образом:

$$\mathbf{y} = H\left(\mathbf{x}, \mathbf{W_H}\right) \cdot T\left(\mathbf{x}, \mathbf{W_T}\right) + \mathbf{x} \cdot \left(1 - T\left(\mathbf{x}, \mathbf{W_T}\right)\right) \qquad (2.10)$$

где x и y - входы и выходы блока соответственно, $\mathbf{H}$ есть основная операция блока с параметрами $\mathbf{W_H}$ и блок переноса (carry) T с

параметрами $\mathbf{W_T}$. Мотивация highway network блока - возможность роста значения градиента по пути переноса carry, способствующая уменьшению вероятность затухания / резкого роста градиента [14].

После этого к последовательности применяется блок двусторонней рекурентной сети GRU[15], позволяющий представить всю данную на вход последовательность, которая может иметь произвольную длину, в виде вектора фиксированной размерности. Главное отличие рекуррентных сетей от остальных - хранение состояния. Рекуррентные сети имеют два входа и два выхода - входы / выходы для состояния модели и входы / выходы для векторов латентного пространства. На первом этапе работы блоку передается первый элемент последовательности на вход для латентных векторов и начальное состояние блока на вход состояния. На втором этапе работы на вход для латентных векторов передается второй элемент входной последовательности, но при этом на вход состояния передается выход состояния блока на предыдущем этапе. На последнем этапе последний элемент входной последовательности обрабатывается рекуррентной сетью с учетом всех предыдущих этапов, и выходной латентный вектор используется как выход всего блока. Подробное устройство блока GRU можно найти в работе авторов[15]. После каждой операции свертки в сети, для улучшения сходимости, используется операция batchnorm[12].

Таким образом определяется блок энкодера, который преобразует наборы символов в вектор фиксированной длины, содержащий внутри некоторое представление входных данных с учетом контекстной информации.

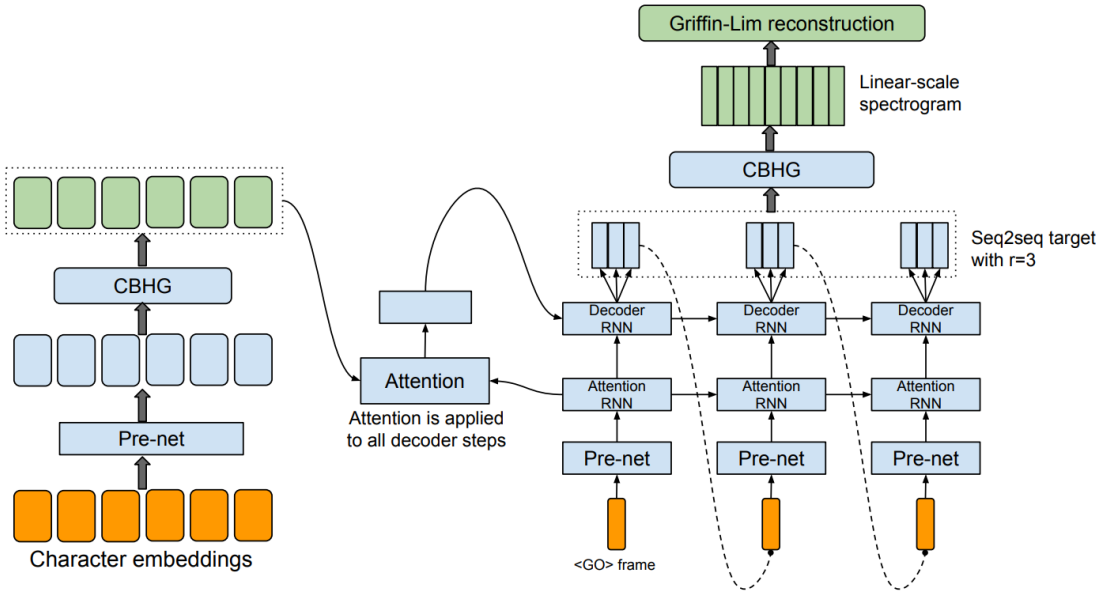


Рисунок 2.2 – Архитектура TTS модели Tacotron

В качестве блока внимания (attantion) использовался блок, основанный на функции tanh [16], который можно описать формулой 2.11:

$$\text{attn}(\mathbf{q}, \text{enc}) = \text{softmax}(\mathbf{V} \cdot \tanh(\mathbf{W} \cdot \mathbf{q} + \text{enc})) \tag{2.11}$$

где $\mathbf{V}, \mathbf{W}$ - матрицы обучаемых параметров, которые кроме того могут быть использованы для того, чтобы размености входного вектора $\mathbf{q}$ и выхода attn совпадали. enc - выход блока энкодера. Блок внимания используется для того, чтобы учесть текстовую информацию во время генерации звукового сигнала. Природа параметра $\mathbf{q}$ (query) зависит от задачи, и в данном случае является состоянием блока декодера.

Блок декодера генерирует выход модели - MEL спектрограмму. В качестве целевых параметров была выбрана MEL спектрограмма с 80

bank фильтрами. Декодер состоит из нелинейного блока, аналогичного с нелинейным блоком в энкодере, блока внимания, двух рекуррентных блоков и блока последнего линейного слоя, который непосредственно предсказывает матрицу спектрограммы и присоединен к последнему рекуррентному блоку.

Работа декодера происходит следующим образом: в первый нелинейный блок декодера на вход на первом этапе передается начальный вектор <go>, вместо которого на следующих этапах используется предыдущее состояние выхода декодера. Нелинейный блок затем передает свой выход в первый рекуррентный блок GRU, выходы которого передаются на вход в блок внимания и рекуррентный блок LSTM[17]; блок LSTM принимает на вход результат операции внимания и выход блока GRU, выдавая на выходе столбцы результирующей MEL-спектрограммы. Так как соседствующие столбцы MEL-спектрограммы сильно коррелируют друг с другом, авторы предлагают генерировать сразу r столбцов матрицы и использовать в качестве состояния декодера последний $r - 1$ столбец. В оригинальной модели гиперпараметр $r = 3$. Стоит заметить, что во время тренировки используется подход Teacher forcing[18], принцип которого заключается в том, что во время тренировки вместо передачи генерированного на прошлом этапе столбца спектрограммы, в модель передается соответствующий столбец из Mel-спектрограммы - желаемого выхода модели, что позволяет упростить и ускорить тренировку рекурентных моделей.

Как видно из рисунка 2.2, после последнего линейного слоя может быть добавлен вспомогательный блок CBHG, однако в работе о клонировании голоса[1] этот блок не используется.

Таким образом, описанная выше архитектура позволяет при помощи

датасета типа текст-речь получить предсказание MEL спектрограммы. Для парамметризации модели на векторах, описывающих спикеров, эти вектора конкатинируются с каждым эмбеддингом выхода энкодера модели Tacotron. Таким образом модель понимает, когда от нее требуется воспроизвести текст выбранным способом. Предполагается, что с большим количеством тренировочных данных модель синтеза речи сможет генерализовать представление спикера и научиться генерировать голос новых, незнакомых, но похожих на спикеров из тренировочной выборки, спикеров.

2.2.3 Задача восстановления звукового сигнала из MEL-спектрограммы

Последний этап генерации голоса на основе тексте - восстановление звукового сигнала на основе MEL-спектрограммы.

Во время трансляции звукового сигнала в MEL спектрограмму методом FFT, часть данных, а именно мнимые коэффициенты ряда Фурье, отвечающие за сдвиг базового сигнала то есть фазу - стираются. Задача заключается в восстановлении закодированного в MEL спектрограмму звукового сигнала в первоначальный базис. У этой задачи есть классический алгоритм решения - алгоритм реконструкции Гриффина-Лимма [19]. Однако на практике алгоритм Гриффина-Лима превносит значительные для слуха артефакты в раскодированный аудио сигнал. Популярный метод решения - использования нейронных сетей. В работе по клонированию [1] для реконструкции фазовой информации спектрограмм используется архитектура waveRNN[21]. Модель waveRNN основывается на своем предшественнике - модели wavenet[**?**], которая моделирует закодированный аудиосигнал $\{x_1, \cdots, x_T\}$ как совместное

распределение:

$$p(\mathbf{x}) = \prod_{t=1}^{T} p\left(x_t \mid x_1, \ldots, x_{t-1}\right) \tag{2.12}$$

В каждом блоке модели используется одномерные конволюции с параметрами dilate $= 1, 2, 4, \cdots, 2^N$. где N - количество блоков в модели. Такой набор параметров позволяет увеличить рецептивное поле модели, не увеличивая при этом количество вычислений[20]. Один блок модели wavenet можно описать следующей формулой:

$$\mathbf{z}(\mathbf{x}) = W_k \cdot \tanh\left(W_{f,k} * \mathbf{x}\right) \odot \sigma\left(W_{g,k} * \mathbf{x}\right) \tag{2.13}$$

где $\mathbf{x}$ - вход слоя, k - индекс слоя, W обучаемые параметры, $*$ - операция свертки, $\odot$ - символ поэлементного умножения, $\cdot$ - символ скалярного умножения, V - матрица локального контекста, $\mathbf{h}$ - вектор глобального контекста. Индекс f символизирует, что параметры относятся к основному пути вычислений модели, и g значит, что веса относятся к дополнительной ветке модели, добавленной для ускорения сходимости тренировки модели. Как утверждают авторы, такая архитектура блока тренируется лучше, чем архитектура без дополнительной ”gate”ветки вычислений[**?**]. Для ускорения обучения каждый блок имеет resudual connection связь, которая складывает результат вычисления блока с входом этого блока:

$$\mathbf{Z}_{\text{skip}}(\mathbf{x}) = \mathbf{z}(\mathbf{x}) + \mathbf{x} \tag{2.14}$$

где + - поэлементное сложение. Кроме того, для ускорения обучения также используется skip connection - выход каждого блока, прежде чем передаваться в последний этап обработки в модели, складывается друг с другом посредством полементного сложения. Послединй этап обработки модели есть парой последовательность операций Relu- одномерная конволюция с ядром размерности 1 и операцией softmax на выходе. Архитектура всей модели изображена на рисунке 2.3.

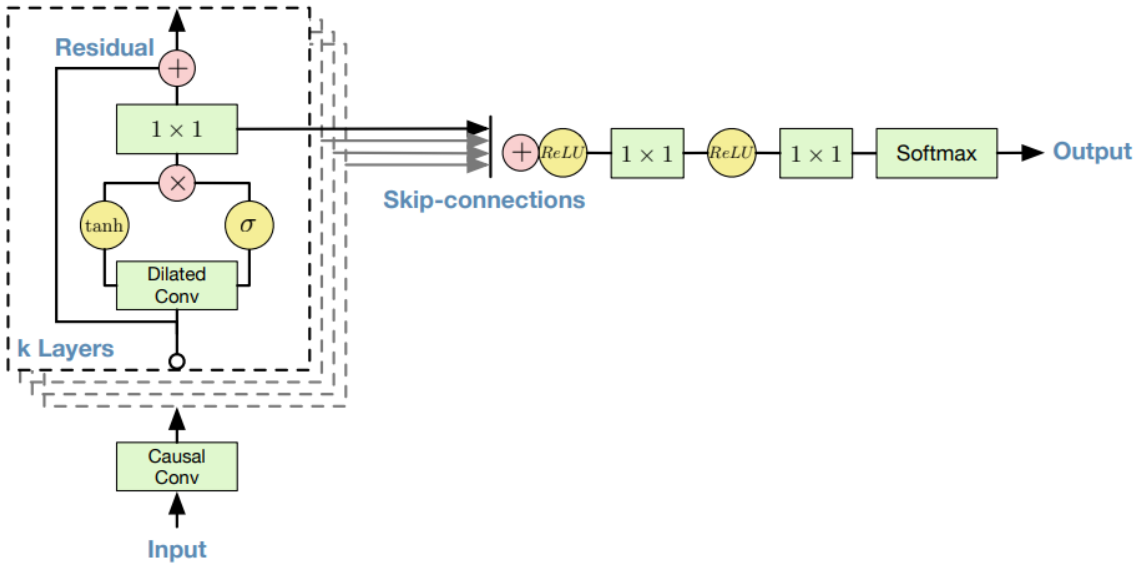


Рисунок 2.3 – Архитектура модели Wavenet

Благодаря своей архитектуре во время тренировки с использованием подхода teacher forcing [18] предсказание каждого элемента выходного сигнала может быть произведено независимо друг от друга, что позволяет тренировать модель в распределенном режиме[20]. Однако, в свою очередь, такой подход замедляет работу модели в режиме инференса: для генерации каждого последующего элемента выхода модели требуется все предыдущие, что не позволяет производить инференс модели в параллельном режиме. Кроме того, полученная архитектура имеет

достаточно высокую вычислительную сложность.

Следующая версия модели - WaweRNN [21] не имеет недостатков, упомянутых выше, благодаря модифицированной архитектуре модели. WaveRNN состоит из одного рекуррентного блока и использует subscaling алгоритм для ускорения генерации выходного сигнала.

Пусть входная спектрограмма $\mathbf{u}$ проецируется в набор элементов $u = P(\mathbf{u})$. Входная последовательность $\mathbf{u}$ разбивается на B подпоследовательностей длиной $\frac{\|\mathbf{u}\|}{B}$. Каждая подпоследовательность с индексом i содержит элементы из $\mathbf{u}$ с индексами $i + kB, k = 0 \cdots \frac{\|\mathbf{u}\|}{B}$. Обозначение u_{i+j} значит, что это элемент j подпоследовательности с индексом i. Тогда алгоритм subscaling можно описать следующей формулой:

$$P(\mathbf{u}) = \prod_{s=0}^{B} \prod_{i=0}^{|\mathbf{u}|/B} P\left(u_{i+s} \mid u_{j+s} \text{ for } j < i \right. \\ \left. u_{k+z} \text{ for } z < s \text{ and } 0 \leq k \leq i + F\right) \tag{2.15}$$

Где гиперпараметр $F \in R$ называется future horizon. Из формулы видно, что каждый элемент u_{i+s} в модели зависит от всех предыдущих значений подпоследовательности с индексом s, в которую входит u_{i+s}. Кроме того, u_{i+s} зависит от элементов подпоследовательностей с индексами $z, z < s$, причем внутренний индекс подпоследовательностей изменяется от 0 до $i + F$, предоставляя для моделирования элемента информацию не только до текущего момента генерации, но и после этого момента на F шагов.

Параллельную генерацию выхода модели можно произвести следующим образом: на первом этапе генерируется F элементов первой из B подпоследовательности. На шаге под номером $F + 1$ возможна параллельная генерация двух выходных элементов: $F + 1$ элемента первой

последовательности и первого элемента второй подпоследовательности. На шаге $2F$ генерируется F элемент второй подпоследовательности, позволяя на шаге $2F + 1$ сгенерировать параллельно элементы $2F + 1$ первой подпоследовательности, $F + 1$ второй подпоследовательности и первый элемент третьей подпоследовательности. Таким образом, на шаге FB будет возможна параллельная генерация сразу B выходных элементов. Такой подход, как утверждают авторы, для небольших значений F $(64, 128)$ позволяет ускорить время инференса в 4 раза[21]. Кроме того, для ускорения модели применялся компрессионный алгоритм прунинга [21], задача которого заключается в уменьшении размерности параметров входящих в сеть операций.

3 Программная реализация

3.1 Данные

Данные представляют собой наборы аудиофайлов, каждый из которых имеет расшифровку сказанного в файле текста и идентификационный номер спикера. Начальные веса модели, обученные на англоязычных аудиозаписях из датасета LibriSpeech [24] и VoxCeleb [25], были взяты у автора статьи Voice Cloning [1]. Для дотренировки использовались следующие датасеты с качественными звукозаписями:

Ruslan: один из самых длительных датасетов на русском языке с одним спикером[22].

M-Ailabs: Обучающие данные состоят из почти тысячи часов аудио и текстовых файлов на разных языках. Для этой работы использовалась только часть датасета с русским языком.

Mozilla Common Voice: Общедоступный набор голосовых данных, созданный на основе голосов добровольцев со всего мира[26]. Содержит аудиозаписи на множестве языков, для обучения была использована только часть датасета на русском языке.

Voxforge + Books: компиляция данных из датасета Voxforge и бесплатных аудиокниг, выполненая автором[30].

Russian single одноголосый датасет для решения задачи генерации текста с ресурса Kaggle[27].

Информация о длительности аудиозаписей каждого датасета и количестве спикеров приведена в таблице 3.1.

Таблица 3.1 – Описание тренировочных датасетов

Название датасета	Количество спикеров	Длительность размеченных аудиозаписей
Ruslan	1	27ч 57мин
M-Ailabs	3	27ч 12мин
Mozilla Common Voice	50	20ч 55мин
Voxforge + Books	206	7ч 2мин
Russian single	1	6ч 29мин
Всего	260	89ч 35мин

Датасеты с плохим качеством звука, несмотря на большое количество данных, могут приводить к появлению артефактов в момент обучения стадии генерации голоса (TTS), поэтому такие датасеты как open tts и open sst [23] во время тренировки использованы не были.

Организация структуры выбранных датасетов не уницифицирована, каждый набор данных хранит разметку и сами аудиофайлы по-разному. Для обучения модели данные должны быть унифицированы, поэтому в рамках работы для каждого датасета был реализован код адаптора для кода препроцессинга аудиозаписей.

3.2 Эксперименты

В качестве основы был использован код репозитория Voice Cloning[1]. Конфигурацию обучения каждого этапа можно наблюдать в таблице 3.2.

Тренировка моделей делится на три отдельных этапа:

– тренировка модели идентификации спикеров;

– тренировка модели условной генерации MEL спектрограмм на основе закодированной информации о спикере и входного текста;

– тренировка модели генерации звукового сигнала на основе MEL спектрограммы с прошлого этапа.

Задача	Модель	Данные	Длительность тренировки
Идентификация спикера	LSTM + GE2E	Voxforge + Audiobooks, Mozilla Rus	1 день 1 GPU
Генерация MEL спектрограмм	Takotron 2	все из таблицы 3.1	5 дней 1 GPU
Генерация аудиофайла	WaveRNN	все из таблицы 3.1	1 день 3 GPU

Таблица 3.2 – Описание этапов тренировки модели Voice Cloning на данных, описанных в главе 3.1

Каждый из этапов требует предварительной подготовки данных для тренировки. Первый этап - тренировка модели идентификации спикеров, для проведения которой необходимо конвертировать аудиозаписи в формат MEL спектрограмм. Проекция векторов, полученных во время обучения на шаге 158370 представлена на рисунке 3.1. Как видно из рисунка, вектора спикеров в проекции визуально отделимы друг от друга, что может указывать на то, что модель научилась отличать речь разных спикеров друг от друга.

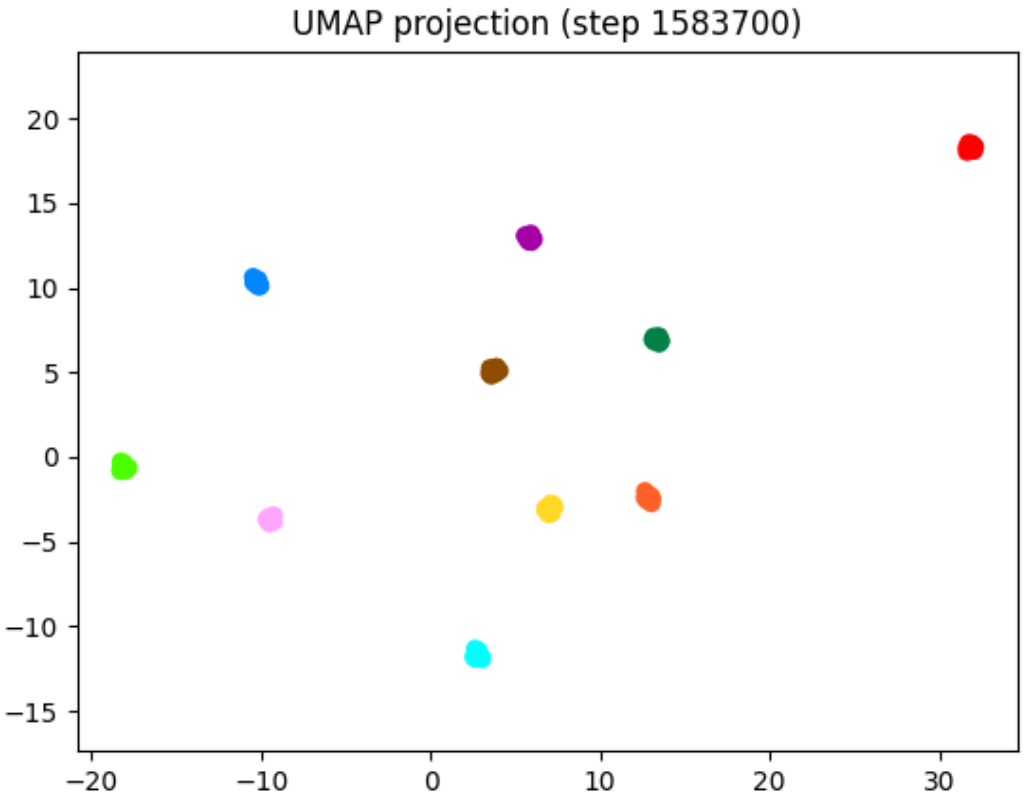


Рисунок 3.1 – Результат работы модели LSTM на обучающих данных на шаге тренировки 158370, спроецированный на плоскость при помощи метода UMAP[28]. Точки одного цвета относятся к аудиозаписям одного спикера

Этап тренировки модели генерации MEL спектрограмм, кроме непосредственно текста для генерации, требует на вход выход модели идентификации спикера. Поэтому перед началом этого этапа для каждой аудиозаписи генерируется соответствующий ему выходной вектор натренированной модели с предыдущего этапа. Этот вектор, в последствии, используется как входной признак для модели генерации MEL спектрограмм. График значений функции ошибки изображен на рисунке 3.3а. Пример спектрограммы, полученный в результате генерации, сопоставленный с соответсвующей спектрограммой из обучающего

набора данных представлена на рисунке 3.2.

Модель генерации звукового сигнала требует на вход MEL спектрограмму, полученную с предыдущего этапа работы модели, поэтому в тренировочный датасет для модели генерации аудио сигнала добавляется еще и выход натренированной модели генерации MEL спектрограмм. График значений функции ошибки во время тренировки изображен на рисунке 3.3б. Как видно из рисунка 3.2, сгенерированная спектрограмма незначительно отличается от оригинальной. Третий этап генерации звуковых сигналов не только решает задачу оценки фазовых значений для блоков спектрограммы, но и работает как фильтр для исправления незначительных дефектов модели генерации.

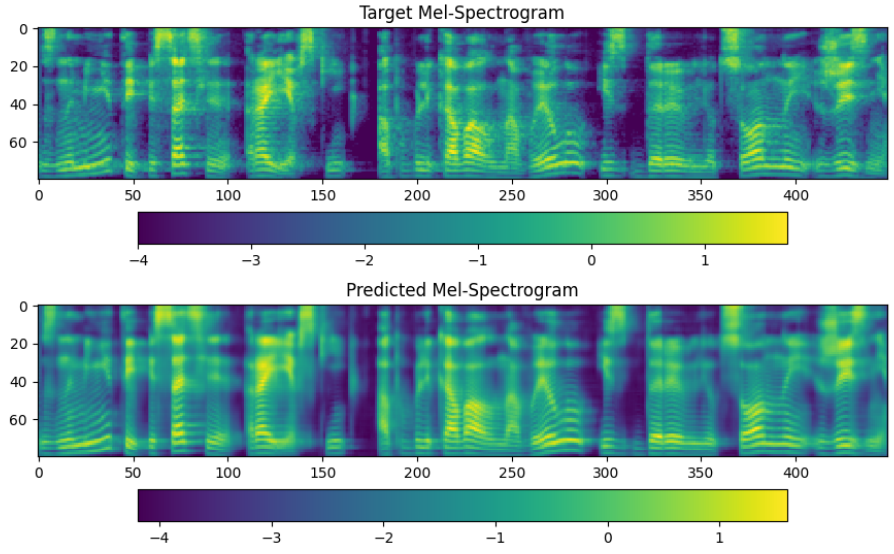
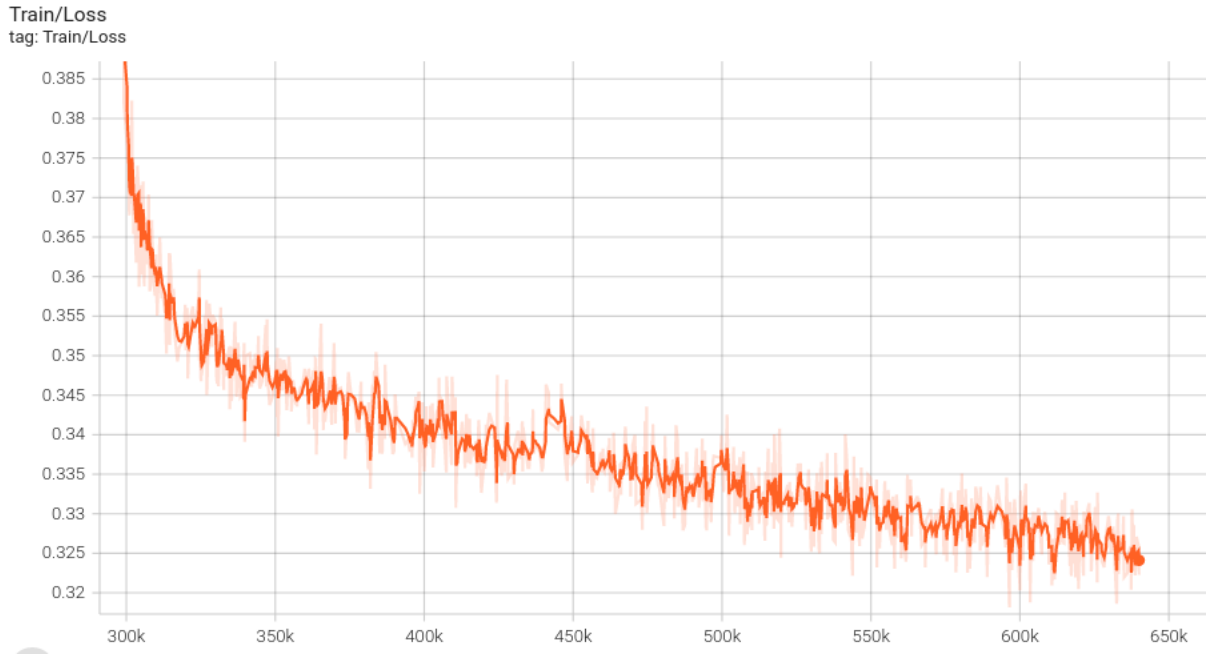
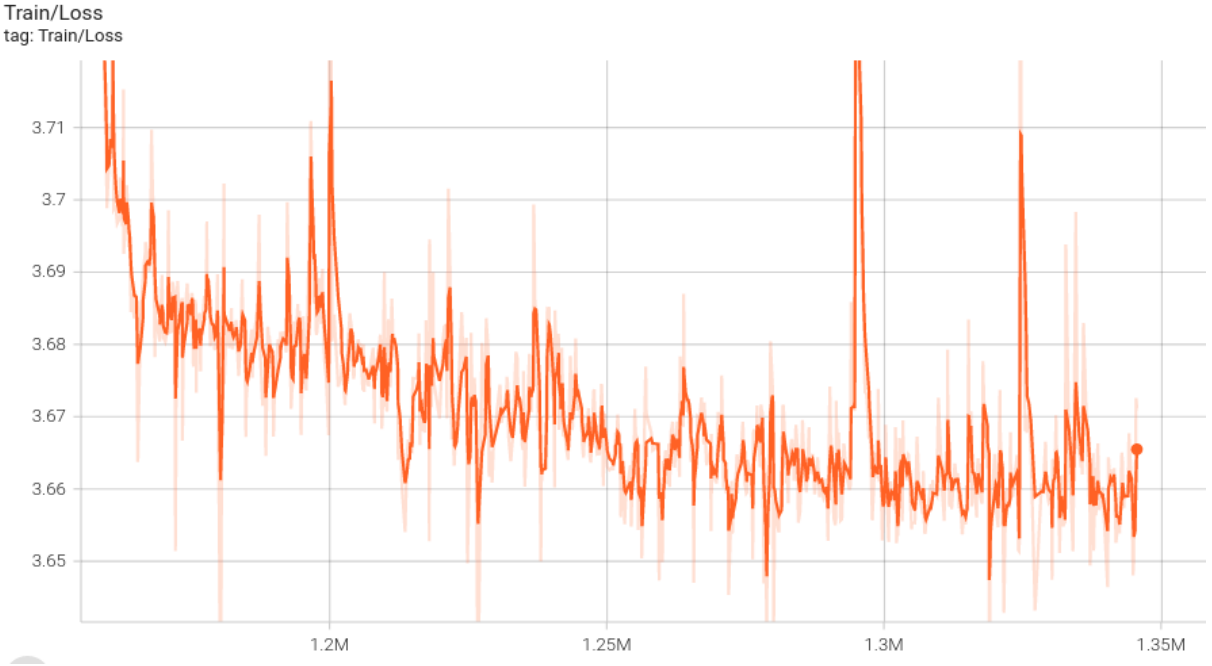


Рисунок 3.2 – Сравнение сгенерированной MEL спектрограммы с соответсвующей спектрограммой из тренировочного датасета на 640000 шаге тренировки модели Tacotron 2



а) График значений функции ошибки Tacotron 2 во время тренировки



б) График значений функции ошибки WaveRNN во время тренировки

Рисунок 3.3 – Графики функций ошибки для моделей Tacotron 2 генерации MEL спектрограмм и WaveRNN генерации аудиофайлов с экспоненциальным усреднением с коэффициентом 0.6

Несмотря на ограниченность данных модель оказалась способна генерировать внятную речь в том случае, когда голос спикера на входящей

аудиозаписи субъективно несколько совпадает с голосом некоторого спикера из тренировочной выборки. Однако в случае, когда субъективно похожего голоса из тренировочных данный не находится - модель не справляется с качественной генерацией длинных текстов, добавляя в вывод различные артефакты. Для улучшения качества генерации на данном этапе была предложена эвристика, описанная в главе 3.3. Главный вывод тренировки - для некоторого множества спикеров модель, несмотря на небольшой объем тренировочных данных, способна генерировать внятную речь.

3.3 Эвристика для улучшения генерации голоса в условиях ограниченности датасета

Тренировочный датасет ограничен, и, как показывате практика, голоса с определенными особенностями (например, высокий женский голос) могут быть не представлены в тренировочной выборке. В результате, пытаясь повторить редкий в тренировочном датасете голос, модель может генерировать некачественный вывод, в котором сложно разобрать слова или присутствует множество артефактов. Однако при клонировании голосов, близких по характеристикам к голосам спикеров из тренировочного датасета, подобные артефакты встречаются редко. Для уменьшения подобных дефектов генерации автором этой работы предложен метод, суть которого заключается в линейном смешивании векторов спикеров с векторами из тренировочного датасета. Под векторами спикеров в этой главе понимаются выходы модели идентификации спикера для соответствующих входов - аудиозаписей спикеров.

Алгоритм устроен следующим образом: в наборе латентных

векторов спикеров из тренировочного датасета находится вектор, максимально близкий по косиноснуму расстоянию к данному вектору. Затем найденный вектор смешивается в некоторой пропорции, в зависимости от желания сохранить больше особенностей речи или сделать её более четкой и понятной, с вектором спикера. Таким образом, если во время тренировки модель ”не слышала”голос, похожий на голос спикера, можно сделать генерацию более похожей на ту, которая присутствует в тренировочном датасете. Формально процедура выше описывается формулой 3.1:

$$\mathbf{x_{mix}}(\mathbf{x}) = m \cdot \mathbf{x} + (1 - m) \cdot \mathbf{x_{c(x)}} \tag{3.1}$$

где $\mathbf{c}$ определяется из формулы 3.2:

$$\mathbf{c}(\mathbf{x}) = \arg\min_i \frac{|\mathbf{x} \cdot \mathbf{x_i}|}{\|\mathbf{x}\|\|\mathbf{x_i}\|} \tag{3.2}$$

где $m \in [0, 1]$ и набор$\{\mathbf{x_i}\}$ - центроиды векторов спикеров из тренировочного датасета, полученых в результате работы модели идентификации спикера. Центроиды могут быть предвычислены заранее, и дополнительное замешивание не потребует большого количества операций во время инференса модели. Параметр m можно подбирать в ручную, однако можно также определить его как зависимым от модуля косинусного расстояния между $\mathbf{x}$ и $\mathbf{x_c}$, например так, как в формуле 3.3:

$$m = m(\mathbf{x}) = 2\sigma(k \cdot \mathbf{x_{c(x)}}) - 1 \tag{3.3}$$

где чем больше коэффициент k - тем сильнее вектор из тренировочного набора будет преобладать над вектором, полученным из входных данных.

Благодаря этой эвристике можно несколько улучшить качество генерации в ущерб уникальным характеристикам входного голоса. В случае, когда вектор входного голоса находится достаточно далеко от всех центроидов тренировочных векторов и сгенерированная речь субъективно неразборчива, применение эвристики помогает получить субъективно лучший результат.

Другой способ улучшить качество генерации - использовать несколько аудиозаписей для генерации латентных векторов спикеров. Как показывает практика, запуская модель идентификации на нескольких разных аудиозаписях одного спикера и усредняя результат, можно получить вектор, содержащий больше полезных характеристик голоса спикера, и, как результат, качество генерации голоса станет субъективно лучше.

3.4 Пользовательский интерфейс

Для удобной работы с моделью клонирования голоса может быть использован графический интерфейс, изображенный на рисунке 3.4, или консольное приложение.

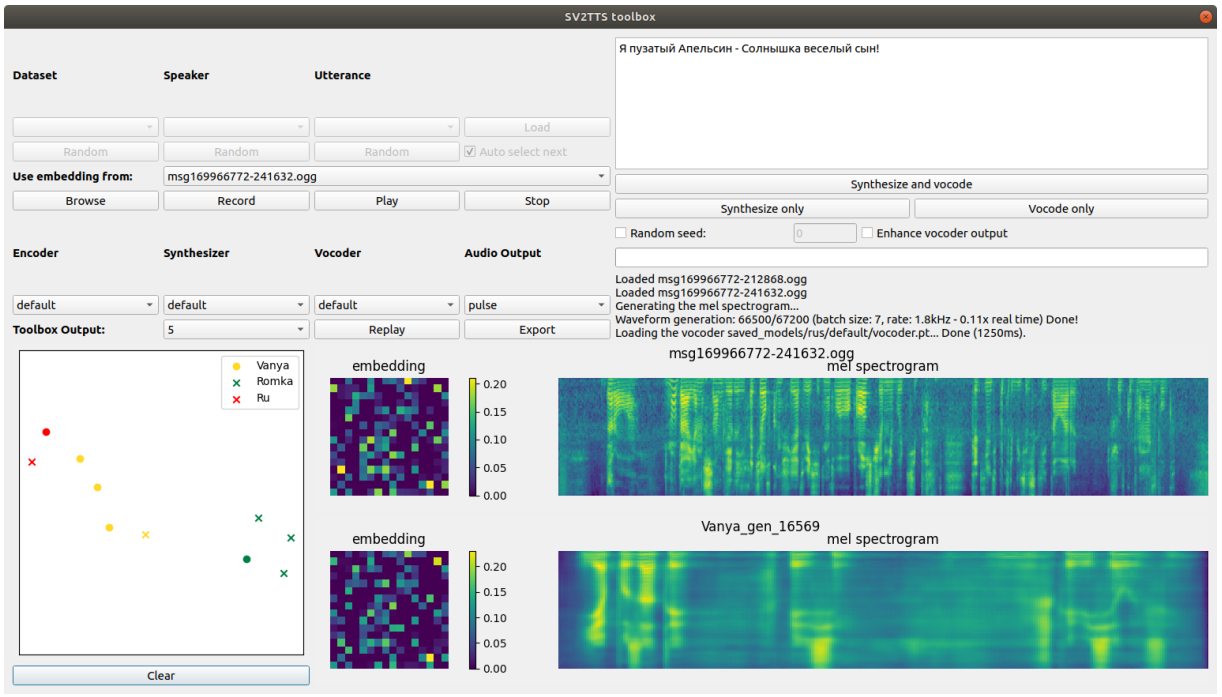


Рисунок 3.4 – Пользовательский интерфейс модели клонирования голоса

Для запуска графического интерфейса необходимо установить зависимости и указать пути до тренировочного датасета и сохраненных весов моделей. Подробную инструкцию по запуску можно найти в описании рабочего репозитория.

Графический интерфейс имеет следующие составляющие: левая верхняя часть отвечает за загрузку аудиозаписей, голос из которых затем можно склонировать, и загрузку весов моделей. В верхнем блоке располагаются кнопки для загрузки данных из датасетов, в нижней - кнопки для загрузки произвольных аудиозаписей. Ниже находится блок выбора весов модели и устройства вывода. В качестве последнего этапа генерации аудиофайлов можно использовать алгоритм Гриффина-Лимма, выбрав соответсвующий пункт в меню Vocoder.

Правая верхняя часть интерфейса отвечает за ввод текста для генерации и непосредсвенно за генерацию звука. Кнопка Synthesize запускает модель генерации MEL-спектрограмм, а кнопка Vocode only

генерирует звуковой сигнал из текущей MEL-спектрограммы. Кнопка synthesize and vocode позволяет получить сразу и MEL-спектрограмму, и аудиофайл.

Нижняя часть интерфейса отвечает за визуализацию: в левом нижнем углу располагается визуализация UMAP проекции[28] латентных векторов загруженных аудиофайлов, полученных при помощи модели идентификации спикера. Разными цветами обозначены разные спикеры, круг означает, что вектор получен из оригинальной дорожки, а крест означает, что вектор получен из сгенерированной аудиодорожки. Правее находятся визуализации спектрограмм и их векторного представления. Верхняя пара изображений относится к входному аудиофайлу, а нижняя - к сгенерированному в текущий момент аудиофайлу.

Для клонирования голоса необходимо сначала загрузить входной аудиофайл при помощи кнопки Browse или записать его с доступного устройства ввода при помощи кнопки Record. Перед генерацией запись можно прослушать, используя для управления кнопки Play и Stop. После загрузки аудиофайла программа автоматически преобразовывает его в MEL сперктрограмму, генерирует латентный вектор, используя модель идентификации спикеров, и визуализирует спектрограмму, латентный вектор и его UMAP проекцию в нижней части интерфейса. После этого необходимо ввести требуемый для генерации текст в окно ввода текста в верхнем правом углу. Используя интерфейс для генерации и выбирая нужные веса для модели, можно получить сгенерированный аудиофайл, для генерации которого использовался введенный ранее текст и латентный вектор загруженного ранее целевого аудиофайла. Визуализации сгенерированного файла автоматически отобразится в нижней части программы. Результат генерации можно прослушать заново, используя

кнопку Replay или сохранить на диск, используюя кнопку Export.

Для генерации нескольких аудиозаписей с разным текстом для множества спикеров удобно использовать консольную программу demo_advanced.py. Программа позволяет использовать текстовый файл в качестве входа, интерпретируя каждую строчку как отдельный текст для генерации. Входные аудиозаписи должны соответствовать следующей структуре папок: аудиозаписи каждого спикера хранятся в отдельной директории, название директории соответствует имени спикера. Как было описано в главе 3.3, генерация, использующая усредненные латентные вектора разных аудиозаписей, работает субъективно лучше, и в консольной программе существует возможность агрегации латентных векторов: для этого необходимо загрузить несколько аудиофайлов в папку спикера и передать в программу параметр, соответствующий режиму агрегации латентных векторов. Кроме того, программу можно запустить в режиме смешения, когда вектора спикеров линейно смешиваются с векторами из тренировочного датасета. Подробное описание алгоритма можно найти в главе 3.3. Подробное описание всех режимов работы можно найти в описании рабочего репозитория.

3.5 Валидация модели клонирования голоса

Для проверки качества работы модели клонирования голоса использовались 10 текстов и 14 спикеров, не входивших в тренировочный набор данных. В качестве метрики оценки качества используется средняя оценка экспертов (Mean Opinion Score) по двум характеристикам: натуральность речи и схожесть сгенерированного голоса с оригинальным. Значения оценок - $1, 2, 3, 4, 5$, где 1 значит низкую оценку характеристики элемента, 5 - высокую оценку характеристики элемента. Для сбора

выбранной метрики автором работы было реализовано веб приложение, код которого находится в директории questionnaire основного репозитория работы.

Результаты сравнения можно наблюдать в таблице 3.3. В сравнении используются генерации из репозитория [30]и генерации этой работы, представленные в двух режимах: генерация без замешивания, но с усреднением латентных векторов, обозначается в таблице как режим 1, и генерация с замешиванием и усреднением латентных векторов, обозначается в таблице стравнения как режим 2.

Режим генерации	Средняя оценка качества генерации	Средняя оценка схожести голосов
Multi Tacotron Voice Cloning [30]	1.93	1.3
Модель этой работы в режиме 1	3.28	**2.69**
Модель этой работы в режиме 2	**3.74**	2.49

Таблица 3.3 – Результат сравнения генераций предшествующей модели и модели этой работы в двух разных режимах генерации

Как видно из таблицы 3.3, в обоих режимах генерации модель, полученная автором этой работы, превосходит предыдущую версию модели как по качеству генерации, так и по оценки схожести оригинальных и сгенерированных голосов. Кроме того из таблицы можно видеть, что эвристика, представленная в главе 3.3, как и предполагалось, улучшает оценку качества генерации уменьшая при этом оценку схожести сгенерированных голосов.

4 Вывод

В результате работы были изучены методы хранения, обработки и преобразования аудиофайлов в цифровом виде. Исследована задача клониравния голоса, включающая в себя такие подзадачи, как задача идентификации спикера, задача генерации MEL-спектрограмм и задача генерации аудиосигнала на основе MEL-спектрограмм. Собраны и предобработаны датасеты для тренировки каждой из подзадач, запущена сама тренировка моделей решения задач клонирования голоса.

В качестве основы была использована кодовая база репозитория[1], однако в процессе работы были выявлены ошибки и недоработки, исправленные автором, улучшен интерфейс логирования статистик тренировки, а также переработаны все фрагменты кода препроцессинга данных, так как предыдущая версия не подразумевала работу с несколькими датасетами, имеющими разную структуру хранения данных. Была выполнена тренировка и последующая валидация моделей для клонирования голоса. В условиях малого набора обучающих данных были разработаны и реализованы эвристики, позволяющие улучшить качество генерации модели клонирования голоса. Произведено сравнение качества генерации обученной модели с конкурирующей версией[30], в результате которого натренированная автором этой работы модель получила оценку выше конкурирующей модели. Для использования модели был подготовлен графический и консольный интерфейс модели клонирования голоса, позволяющий пользователям клонировать произвольный голос, визуализировать латентные вектора и MEL-спектрораммы и подбирать коэффициенты для эвристик улучшения генерации.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Jia, Ye and Zhang, Yu and Weiss, Ron J. and Wang, Quan and Shen, Jonathan and Ren, Fei and Chen, Zhifeng and Nguyen, Patrick and Pang, Ruoming and Moreno, Ignacio Lopez and Wu, Yonghui: Transfer Learning from Speaker Verification to Multispeaker Text-To-Speech Synthesis. 2018. arXiv 1806.04558

2 Josh Beggsl, Dylan Thede: Designing Web Audio. ISBN 1-56592-353-7 – M: O'REILLY, 2001. Источник – docstore.mik.ua/orelly/web2/audio/index.html

3 Gonzalez R.C., Woods R.E. Processing digital Image 4 global edition – M: Pearson, 2017. – 993c.

4 Kay van Najarian, Robert Splinter: Biomedical SignalandImage Processing. 2012. Taylor & Francis Group, DOI 10.1201/b11978. – 441 c.

5 Allen B. Downey: Think DSP – M: Green Tea Press, 2014. – 153 c.

6 Brian McFee, Colin Raffel, Dawen Liang, Daniel, Patrick Whittlesey Ellis, Matt McVicar, Eric Battenberg, Oriol Nieto: Librosa: Audio and music signal analysis in Python 14th PYTHON IN SCIENCE CONF. (SCIPY 2015) – 2015.

7 S. B. Davis and P. Mermelstein: Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentence, IEEE, Trans. Acoust. Speech Signal Processing, том 28, №4, с. 357–366, 1980.

8 M., Laxmi Narayana and Kopparapu, Sunil Kumar: Choice of Mel Filter Bank in Computing MFCC of a Resampled Speech. 2014. arXiv 1410.6903

9 Sepp Hochreiter and Jurgen Schmidhuber: Long short-term memory. Neural computation, том. 9, №8, с. 1735–1780, 1997.

10 Wan, Li and Wang, Quan and Papir, Alan and Moreno, Ignacio Lopez: Generalized End-to-End Loss for Speaker Verification. 2017. arXiv 1710.10467

11 Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J. Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, Quoc Le, Yannis Agiomyrgiannakis, Rob Clark, Rif A. Saurous: Tacotron: Towards End-to-End Speech Synthesis. 2017. arXiv 1703.10135

12 Paszke, Adam and Gross, Sam and Massa, Francisco and Lerer, Adam and Bradbury, James and Chanan, Gregory and Killeen, Trevor and Lin, Zeming and Gimelshein, Natalia and Antiga, Luca and Desmaison, Alban and Köpf, Andreas and Yang, Edward and DeVito, Zach and Raison, Martin and Tejani, Alykhan and Chilamkurthy, Sasank and Steiner, Benoit and Fang, Lu and Bai, Junjie and Chintala, Soumith: PyTorch: An Imperative Style, High-Performance Deep Learning Library. 2019. arXiv 1912.01703

13 He, Kaiming and Zhang, Xiangyu and Ren, Shaoqing and Sun, Jian: Deep Residual Learning for Image Recognition. 2015. arXiv 1512.03385

14 Rupesh Kumar Srivastava, Klaus Greff, Jürgen Schmidhuber: Highway Networks. 2015. arXiv 1505.00387

15 Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on equence modeling. 2014. arXiv:1412.3555

16 Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton: Grammar as a foreign language. In Advances in Neural Information Processing Systems, c. 2773– 2781, 2015.

17 Staudemeyer, Ralf C. and Morris, Eric Rothstein: Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks. 2019. arXiv 1909.09586

18 John F. Kolen; Stefan C. Kremer :A Field Guide to Dynamical Recurrent Networks. John Wiley Sons. c. 202–. ISBN 978-0-7803-5369-5. 2001

19 D. Griffin, J. Lim: Signal estimation from modified shorttime Fourier transform. IEEE Trans. Acoust., Speech, Signal Process., Том 32, №2, с. 236–243. 1984.

20 Oord, Aaron van den and Dieleman, Sander and Zen, Heiga and Simonyan, Karen and Vinyals, Oriol and Graves, Alex and Kalchbrenner, Nal and Senior, Andrew and Kavukcuoglu, Koray: WaveNet: A Generative Model for Raw Audio. 2016. arXiv 1609.03499

21 Kalchbrenner, Nal and Elsen, Erich and Simonyan, Karen and Noury, Seb and Casagrande, Norman and Lockhart, Edward and Stimberg, Florian and Oord, Aaron van den and Dieleman, Sander, Kavukcuoglu, Koray: Efficient Neural Audio Synthesis. 2018. arXiv 1802.08435

22 Gabdrakhmanov L., Garaev R., Razinkov E. (2019) RUSLAN: Russian Spoken Language Corpus for Speech Synthesis. In: Salah A., Karpov A., Potapova R. (eds) Speech and Computer. SPECOM 2019. Lecture Notes in Computer Science, vol 11658. Springer, Cham

23 Russian Open Speech To Text (STT/ASR) Dataset : сайт репозитория. – URL: https://github.com/snakers4/open_stt (дата обращения: 10.05.2022).

24 LibriSpeech ASR corpus: официальный сайт. – URL: http://www.openslr.org/12/ (дата обращения: 10.05.2022).

25 A. Nagrani, J. S. Chung, W. Xie, A. Zisserman: Voxceleb: Large-scale speaker verification in the wild. 2019. – URL: https://www.sciencedirect.com/science/article/pii/S0885230819302712. Computer Science and Language.

26 Mozilla Common Voice Dataset: оффициальный сайт. – URL: https://commonvoice.mozilla.org/en/datasets (дата обращения: 10.05.2022).

27 Russian Single Speaker Speech Dataset: страница датасета. – URL: https://www.kaggle.com/datasets/bryanpark/russian-single-speaker-speech-dataset (дата обращения: 10.05.2022).

28 McInnes, Leland and Healy, John and Melville, James: UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. 2018. arXiv 1802.03426

29 Tacotron Voice Cloning Rus: сайт репозитория. URL - https://github.com/dupeljan/Real-Time-Voice-Cloning

30 Multi-Tacotron Voice Cloning (Russian/English): сайт репозитория. URL – https://github.com/vlomme/Multi-Tacotron-Voice-Cloning (дата обращения: 10.05.2022).