Проект по КТ 4 семестр 2020

Анализ методов очистки аудиозаписей при помощи нейронной сети с различной архитектурой

Удовин Илья, Щербаков Андрей
группа Б05-874, ФУПМ, МФТИ
https://github.com/Andrey-Shcherbakov/Project_4term

Цели проекта совместно с ТЗ:

- Научиться работать с аудиофайлами и понять что такое шум с физической точки зрения
- Научиться распознавать различные типы шумов в аудиофайлах с помощью машинного обучения
- Проанализировать частотность встречаемости различных шумов в аудио файлах для дальнейшей концентрации внимания
- Изучить стандартные алгоритмы очистки аудио, их реализацию на языке Python и принцип работы с физической точки зрения
- Проанализировать эффективность алгоритмов очистки против различных типов шумов

Содержание

Внимание! В данном отчете рассматриваются конкретные типы шумов, которые мы и будем анализировать. Сами типы были выбраны по 2 критериям:

1. Субъективная частотность встречаемости (т.к. как нам казалось, опираясь на персональный опыт)

2. Наличие базы данных, содержащей примеры этого шума

1 Физика шума и алгоритмов очистки

В данной работе аудиофайл рассматривается как спектр, зависящий от времени. Чаще всего применяется упрощение – представляется с помощью спектральной центроиды от времени, где центральная спектроида, это:

$$centroid = \frac{\sum_{n=0}^{N-1} f(n) * x(n)}{\sum_{n=0}^{N-1} x(n)}, \tag{1}$$

где x – амплитуда спектра , f – частота (сам спектр имеет вид (x, f)). Кроме того, очень тихие записи мы автоматически усиливаем и в каждой записи обрезаем моменты долгой тишины (с помощью функции `librosa.effects.trim()`)

2 Срез слишком громких и слишком тихих звуков (Noise reduction using power)

При записывании аудиозаписи могут происходить очень громкие и очень тихие шумы, мешающие восприятию записи. Для уменьшения этого эффекта приведем аудиозапись к одной громкости (по сути сгладим шумы): громкие составляющие сделаем тише а тихие немного громче, причем совсем громкие и совсем тихие обрезаем. Для этого использовалась функция `AudioEffectsChain()` из библиотеки `python_speech_features`, плюс мы округляли спектр, т.к. он представлен в электронном виде, поэтому возникал дополнительный шум, связанный с неточностью представления. Минусы: может усилить слабый шум и меньше передает громкость голоса.

3 Срез слишком больших пиков центроиды (Noise reduction using centroid analysis)

При записывании аудиозаписи могут происходить громкие шумы, мешающие восприятию записи, причем громкие резкие шумы сильно выделяются на остальном фоне – centroid становится сильно выше, чем на остальной записи. Используя это свойство мы можем выявлять места такого шума и фильтровать в них составляющие имеющие максимальные амплитуды (максимальную громкость, что и является шумом). Для этого использовалась функция

`AudioEffectsChain()` из библиотеки `python_speech_features` и методы библиотеки `numpy`. (`reduce_noise_centroid_s()` (from simple))

Ещё один способ найти такие выделяющиеся резкие шумы – это не просто их отфильтровать, как сделано выше, но затем ещё и убрать менее очевидные резко выделяющиеся по своей форме шумы. В нашем коде это делается с помощи растягивания и сжимания спектра с округлением вниз (таким образом отсекаются очень большие и очень мелкие шумы) (`reduce_noise_centroid_mb()` (from math boundaries))

4 Нормализация Мел-частотных кепстральных коэффициентов (Noise reduction using MFCC (el-frequency cepstral coefficients))

Мел – единица высоты звука, основанная на восприятии этого звука нашими органами слуха. Как известно, АЧХ человеческого уха даже отдаленно не напоминает прямую, и амплитуда – не совсем точная мера громкости звука, а восприятие частоты тоже не линейное. Восприятие наиболее близко к логарифмическому с формулой $m = 1125 \ln(1 + f/700)$, где m – это мел, а f – частота. Кепстр – функция обратного преобразования Фурье от логарифма спектра мощности сигнала, по своим свойствам он хорошо подходит для анализа речевого аппарата человека и звуков, которые он создаёт. Сами мел-кепстральные коэффициенты – это кепстр (только не преобразования Фурье, а дискретно-косинусного преобразования) от спектра сигнала на мел-графике (приведенные частоты-амплитуды) (получаем при помощи функций `python_speech_features.base.mfcc()` и `python_speech_features.base.logfbank()`)

Значения MFCC не очень устойчивы в присутствии аддитивного шума, и поэтому обычно нормализуют их значения, чтобы уменьшить влияние шума (отсечь то, что не походит на человеческую речь). Некоторые исследователи предлагают модификации базового алгоритма MFCC для повышения надежности, например, путем увеличения амплитуд логарифмической плавности до подходящей мощности (около 2 или 3) перед выполнением DCT (дискретное косинусное преобразование) (выполнено функцией `python_speech_features.base.lifter()`)), что уменьшает влияние компонент с низкой энергией (тихих шумов).

С помощью суммы квадратов находим наиболее отстоящий от нужного MFCC, переводим его обратно в стандартный спектр (Герцы) и обрезаем. Причем можем обрезать как снизу (остаются более высокие звуки), так и сверху (остаются более низкие). Теорию подробнее можно найти в источниках [1] и [2].

5 Очистка медианным фильтром (Noise reduction using median)

Довольно распространенный фильтр. Основан на том, что импульсные помехи – это кратковременные одиночные импульсы с высокой амплитудой и их можно устранить, пробегая по значениям и сравнивая соседние.

Значения отсчётов внутри окна фильтра сортируются в порядке возрастания (убывания), и значение, находящееся в середине упорядоченного списка, поступает на выход фильтра. В случае чётного числа отсчётов в окне выходное значение фильтра равно среднему значению двух отсчётов в середине упорядоченного списка. Окно перемещается вдоль фильтруемого сигнала и вычисления повторяются.

Реализован с помощью встроенной в библиотеке `soundfile` функции `sp.signal.medfilt()`.

6 Задача классификации

Одна из самых распространенных задач в машинном обучении – задача классификации. Цель состоит в том, чтобы сопоставить данному объекту элемент из конечного множества классов, к которому этот объект должен принадлежать. Для численной параметризации задачи классификации используется вектор со значениями от 0 до 1, где значение k-го элемента отображает степень уверенности в том, что объект принадлежит k-му классу. Искомому классу соответствует максимум из этих значений.

7 Источники шумов

В качестве обучающего набора данных был взят UrbanSound8K[1]. Он содержит более 8 тысяч записей длительностью в несколько секунд, на которых присутствуют распространенные звуки городских шумов, а именно:

[1]Набор можно найти, например, на kaggle.com

1. шум кондиционера
2. сигнал автомобиля
3. крики детей
4. лай собаки
5. звук дрели
6. звук двигателя автомобиля
7. звук выстрела
8. звуки стройки
9. звук сирены
10. уличная музыка

8 Рекуррентные нейронные сети

В отличие от нейронных сетей с прямой связью, рекуррентные сети могут использовать свою внутреннюю память для обработки последовательностей произвольной длины (рис. 1). Поэтому данные сети применимы в задаче, где данные разбиты на части и представляют собой большой массив численных значений. Анализ аудиоданных как раз представляет собой такую задачу.

Среди рекуррентных нейронных сетей наиболее распространены сверточные нейронные сети (convolutional neural networks), а также сети с долгой краткосрочной памятью.

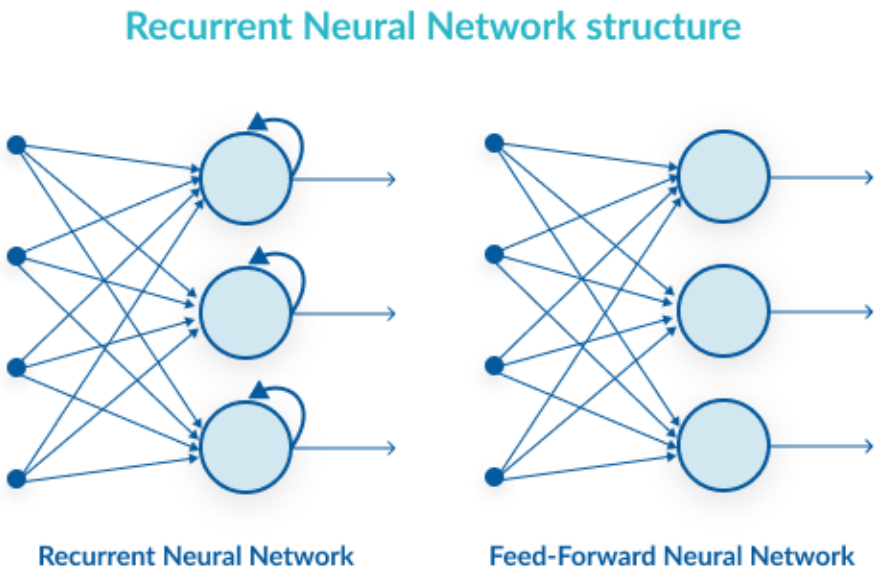

Рис. 1: Схема рекуррентной нейронной сети и нейронной сети с прямой связью.

9 LSTM (Long short-term memory)

В данной работе было использовано несколько моделей нейронных сетей, а именно одно- и двухмерная сверточная модель и LSTM, т.е. с долгой краткосрочной памятью.

9.1 Сравнение моделей

В качестве целевой метрики была взята *точность* (*accuracy*), которая определяется как доля правильных ответов. Если изобразить на графике зависимость точности от раунда обучения, то чем ближе полученная кривая будет к левому верхнему углу графика, тем точнее будет модель. Диагональный вид графика соответствует случайному угадыванию.

Как можно видеть на рис. 2, все использованные модели показывают хорошую точность. Хотя сверточные сети почти меньше переобучаются, их итоговая точность чуть ниже, чем у LSTM (см. таблицу 1). Именно точностью и обоснован выбор ее в качестве основной модели.

модель	точность
Conv 1D	0,88
Conv 2D	0,92
LSTM	0,94

Таблица 1: Усредненные значения точности для трех моделей после 30 раундов обучения (более подробные результаты в папке logs).

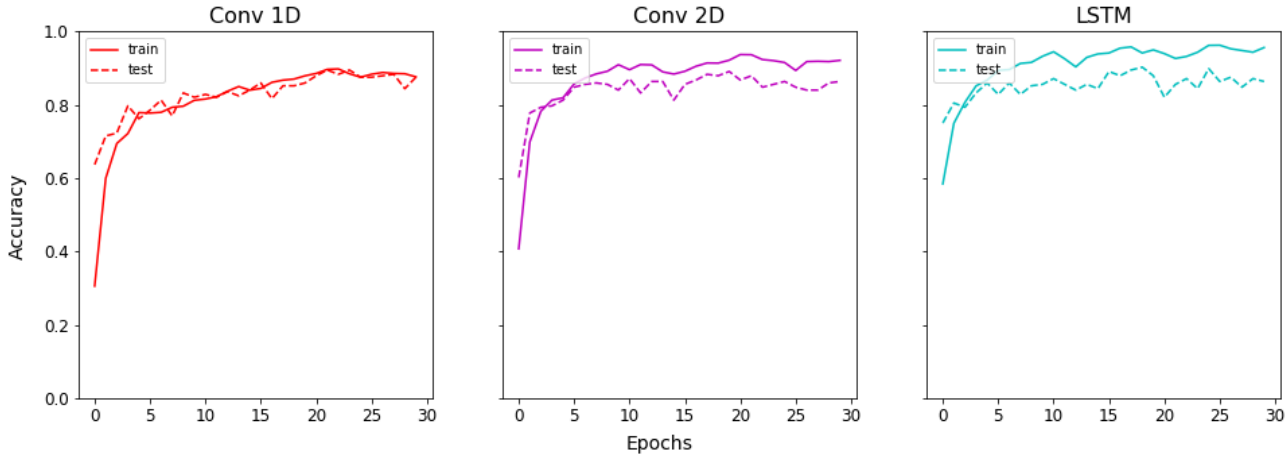

Рис. 2: Зависимость точности от раунда обучения на обучающей (сплошная линия) и тестовой (пунктирная линия) выборках для использованных в работе моделей: (слева направо) одномерная сверточная модель, двухмерная сверточная модель, LSTM-модель.

Список литературы

[1] Dr. Amita Dev, Poonam Bansal – *Robust Features for Noisy Speech Recognition using MFCC Computation from Magnitude Spectrum of Higher Order Autocorrelation Coefficients*. International Journal of Computer Applications (0975 – 8887) Volume 10– No.8, November 2010 [.pdf]

[2] Sourabh Ravindran, David V.Anderson, Malcolm Slaney – *Improving the Noise-Robustness of Mel-Frequency Cepstral Coefficients for Speech Processing*. [.pdf]