МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра информатики и программирования

РЕШЕНИЕ ЗАДАЧИ РАСПОЗНАВАНИЯ АККОРДОВ ИЗ АУДИОСИГНАЛА С ПОМОЩЬЮ ГЛУБОКОГО ОБУЧЕНИЯ

АВТОРЕФЕРАТ МАГИСТЕРСКОЙ РАБОТЫ

студента 2 курса 273 группы
направления 02.04.03 — Математическое обеспечение и администрирование
информационных систем
факультета КНиИТ
Путилова Даниила Ивановича

Научный руководитель
к.э.н., доцент _____ Л. В. Кабанова

Заведующий кафедрой
к.ф.-м.н., доцент _____ М. В. Огнева

Саратов 2023

ВВЕДЕНИЕ

Распознавание аккордов — это задача автоматического определения состава аккорда на основе аудиосигнала. Это важная задача в области музыкальной информатики, которая используется для различных целей, таких как анализ музыкальных композиций, поиск музыки по жанру или тональности, создание музыкальных рекомендательных систем и т.д.

Благодаря последовательным и частотным характеристикам, музыку можно анализировать.

Целью настоящей магистерской работы является проектирование и реализация модели, способной распознавать последовательности аккордов из полученного аудиосигнала.

В соответствии с поставленной целью необходимо решить следующие задачи:

- обзор существующих методов и технологий для распознавания последовательностей;
- анализ актуальных решений в сфере распознавания последовательностей аккордов;
- сравнение существующего программного обеспечения;
- исследование эффективности различных алгоритмов распознавания аккордов;
- обучение нескольких моделей с использованием различных методов машинного обучения;
- реализация пайплайна для практического использования модели.

Методологические основы распознавания аккордов из аудиосигнала с помощью аудиосигнала с помощью глубокого обучения представлены в работах Набиуллина Д.А. [1], Корженовского Ф. [2], Мюллера А. [3].

Практическая значимость магистерской работы. Практическая значимость работы заключается в создании программного продукта для распознавания последовательностей аккордов из аудиофайлов, которые могут использоваться как начинающими музыкантами, так и любителями для игры или анализа композиций. Разработанная модель предоставляет более универсальный подход к распознаванию, отличный от существующих аналогов.

Структура и объём работы. Магистерская работа состоит из введения, 3 разделов, заключения, списка использованных источников и 5 приложений.

Общий объем работы – 82 страниц, из них 55 страницы – основное содержание, включая 45 рисунков, 27 страниц – приложения, список использованных источников информации – 32 наименований.

ОСНОВНОЕ СОДЕРЖАНИЕ РАБОТЫ

Машинное обучение и анализ данных — это в основном итерационные процессы, в которых данные задают ход анализа. Крайне важно для этих процессов иметь инструменты, которые позволяют оперативно и легко работать

С помощью базы Kaggle [4] находим наиболее подходящий для нашей задачи набор аудио-файлов в которых записаны аккорды. В данном случае был выбран «Piano Triads Wavset» содержащий 432 аудиофайла формата $.wav$. В датасете записаны все основные виды аккордов: Major, Minor и Diminished. Каждый аккорд записан в 6 разных октавах и в формах обычной и первой инверсии. Все сэмплы 32-битны и имеют частоту 44100Гц.

Ранее было сказано, что для распознавания аккордов часто используется представление аудиосигналов в виде хромаграммы. Она представляет собой временной ряд хромовекторов, которые представляют гармонический контент в определенное время в аудио, как $c \in \mathbb{R}^{12}$. Каждый c^i обозначает тон, а его значение указывает текущую значимость соответствующего класса тона. Хромовекторы вычисляются с помощью применения набора фильтров к частотно-временному представлению аудио. Это представление является результатом либо кратковременного преобразования Фурье (STFT), либо преобразования с постоянной q (CQT), мы воспользуемся последним из-за лучшего частотного разрешения в области более низких частот.

С помощью функции hpss разложим временной аудиоряд на гармонические и перкуссионные компоненты. Эта функция автоматизирует преобразования STFT->HPSS->ISTFT и гарантирует, что выходные сигналы имеют равную длину входному сигналу y. STFT (Short-time) представляет сигнал в спектрограмму путем вычисления дискретного преобразований Фурье (DFT) [1] по коротким пересекающимся окнам, затем HPSS (Harmonic Percussion Sound Separation) выделяет гармонические и перкусионные элементы, а ISTFT (Inverse) преобразует данные обратно во временной ряд.

Используем преобразование постоянной Q для гармонической компоненты [5], и с помощью функции chroma_cqt получим хромаграмму сигнала. Каждую хромаграмму сохраняем в папке своего класса в формате $.png$.

Далее, приступим к импорту нейронной сети DenseNet. Для всей работы с моделями нейронными сетей (построением, обучением, вычислением) мы будем использовать API Keras.

Зададим последние слои вручную. Выходной слой имеет 36 нейронов для 36 классов и функцию активации $softmax$ для вывода вероятностных результатов распознавания для каждого класса.

Для эффективного обучения нейронных сетей практически всегда рекомендуется выполнять некоторое масштабирование входных значений. Мы можем нормализовать значения пикселей в диапазоне 0 и 1, разделив каждое значение на максимальное [6]. Большинство алгоритмов обучения ожидают числовые значения для лучшей эффективности. Для этого кодируем метки ($labels$) как двоичные вектора с помощью LabelBinarizer [7] из пакета scikit-learn, т.к. нейронные сети могут иногда находить определенные порядки и иерархии в числовых значения меток.

Для оценки эффективности модели, мы предъявляем ей новые размеченные данные (размеченные данные, которые она не видела раньше). Обычно это делается путем разбиения собранных размеченных данных на две части. Одна часть данных используется для построения нашей модели машинного обучения и называется обучающими данными (training data) или обучающей выборкой. Остальные данные будут использованы для оценки качества модели, их называют тестовыми данными (test data), тестовой выборкой.

В библиотеке scikit-learn [8] есть функция train_test_split, которая перемешивает набор данных и разбивает его на две части. Эта функция отбирает в обучающий набор 80% строк данных с соответствующими метками. Оставшиеся 20% данных с метками объявляются тестовым набором.

Когда разделяющая плоскость слишком точно описывает классы из обучающей выборки и теряется обобщающая способность нейронная сети, такой эффект называется переобучением. Чтобы контролировать появление данного эффекта в процессе обучения обучающая выборка разбивается еще на две: обучающую и валидационную (val). Затем в процессе анализа процесса обучения нейронной сети, мы сможем заметить переобучение, если точность предсказаний на валидационной выборке начнет падать или, стоять на месте, пока точность обучающей выборки улучшается.

Задаем функцию автоматического замедления темпа обучения, как толь-

ко оно начинает стагнировать. ReduceLROnPlateau будет следить за валидационной точностью модели (valaccuracy) и если в течении 5 эпох не будет происходить изменений, то темп обучения будет умножен на factor. Это нужно, чтобы алгоритм смог добраться минимума с помощью более меньших шагов.

Наконец, обучаем модель с помощью команды fit. Указываем xtrain, ytrain как обучающую выборку. Параметр steps_per_epoch отвечает за количество сэмплов загружаемых в модель за одну эпоху (epoch), то есть итерацию обучения. Выбираем валидационную выборку в параметре validation data и функцию уменьшения темпа обучения в callbacks.

```
Всего-тест; 87 верно-предсказано: 25 неверно-предсказано:  62
Точность: 28.736 %
```

Результатом на тестовой выборке стала точность модели — 28.736%.

Судя по высокой точности обучения и лишь слегка увеличивающейся валидационной точности, можно сказать, что модель сильно переобучилась в силу своей сложной архитектуры. Комплексная архитектура $DenseNet$ оказала только негативное влияние при достаточно простых входных данных как в этой задаче.

Модель типа Sequential — это простейший вид моделей в Keras для нейронных сетей, которые построены из одного набора слоев соединенных последовательно. В начало модель поместим два сверточных слоя. Следующий слой Dropout отключает случайные 25% нейронов каждую итерацию работы модели, чтобы ослабить эффект переобучения. Далее — слой, который преобразует данные двумерной матрицы в вектор, называемый Flatten. Он позволяет обрабатывать выходные данные стандартными полносвязными слоями. Выходной полносвязный слой имеет функцию softmax и 36 как количество классов.

Наконец, обучаем модель с помощью команды fit, аналогично.

По процессу обучения модели можно заметить, что за 15 эпох валидационная точность улчшилась с 0.1954 до 0.82. Теперь оценим модель на тестовых данных, используя $predict$.

```
Всего-тест; 87 верно-предсказано: 81 неверно-предсказано:  6
Точность: 93.103 %
```

Результатом на тестовой выборке стала точность модели — 93.103%.

Модель два раза ошиблась на аккордах одного типа находящихся в полутоне друг от друга (G# min, G min), и 4 раза неправильно определила уменьшенный аккорд. Это может быть вызвано как недостаточной чистотой исходных данных, так и тем фактом, что некоторые уменьшенные аккорды по своему гармоническому построению отличаются от мажорных или минорных только по одному тону.

Использование модели происходит следующим образом:

- wav информация песни загружается с помощью librosa;
- методом скользящего окна считывается хромаграмма временного отрезка;
- хромаграмма преобразуется в изображение и отправляется на вход модели;
- полученные предсказания по одному сохраняются в массив.

Для обучения следующих моделей был использован датасет The McGill Billboard Project [9]. Датасет содержит около 740 песен из чарта Billboard, размеченных по стандартам распознавания аккордов. Каждая песня разбита на временные отрезки с определенным аккордом. Также в датасете присутствует данные о тюнинге, тональности песни.

Всего хромаграмма состоит из 24 столбцов, 12 из которых отвечают за средний частотный спектр, другие 12 - за басовый спектр. Эти данные будут использованы как основные признаки для моделей, так как содержат важную для распознавания гармоническую информацию.

Данные требуют предварительной обработки. Так как аккордовая разметка имеет не соответствует хромаграммам по временным отрезкам, с помощью скользящего окна (были использованы разные размеры и шаг) хромаграммы усредняются и в соответствие им ставится метка аккорда. Затем, для лучшего понимания различными моделями, значения хромаграмм нормализуются в диапазон [0,1].

Датасет предлагает несколько вариантов разметки аккордов: только мажор-минор, мажор-минор и септаккорды, мажор-минор с инверсиями и полный вариант со всеми возможными вариантами расширений аккордов. В обучении в основном будет использован вариант мажор-минор с септаккордами, но и другие варианты тоже будут задействованы в зависимости от качества

полученных результатов.

Для облегчения задачи классификации приводим названия аккордов к единому формату (только бемоли), что позволяет избавиться от ненужных классов.

Для данного эксперимента используем фреймворк Pytorch. Он предоставляет инструменты для проектирования и реализации нейронных сетей любого масштаба и сложности. С помощью фиксированных модулей пользователь комбинирует заранее определённые блоки в граф вычислений.

Все данные представляются в виде набора последовательностей признаков и аккордов (песен). Так как рекуррентные сети работают только с последовательностями фиксированной длины, используем паддинг и усечение. Задаем максимальную длину последовательностей 600, длина большинства последовательностей меньше этого порога.

После разделения данных на тренировочные выборки, задаем архитектуру модели. В Pytorch необходимо проектировать пайплайн прохода данных через слои модели, получение результатов и их потери.

Используем слой сети с долгой кратковременной памятью (LSTM) как основной. Задаем размерность входных последовательностей, количество слоев (наложенных друг на друга) и двунаправленность.

В задачах, где доступны все временные шаги входной последовательности, двунаправленные LSTM обучают два вместо одного LSTM во входной последовательности. Первый на входной последовательности как есть, а второй на обратной копии входной последовательности. Это может обеспечить дополнительный контекст для сети и привести к более быстрому и даже более полному изучению проблемы.

На выходе модели полносвязный слой, классифицирующий выход LSTM на некое количество классов аккордов соответственно.

Для обучения модели выбраны следующие параметры: оптимизатор Adam с начальным шагом обучения 0.01, функция потери CrossEntropyLoss, планировщик ReduceLROnPlateau уменьшающий шаг обучения в случае стагнации градиентного спуска.

В результате обучения, модель показала точность 43% на мажор-минор и септаккордах.

Чтобы улучшить результат можно попробовать другой вариант LSTM

сети. Такая LSTM называется Stateful, в процессе обучения у такой сети не обнуляются скрытые состояния и состояния ячеек, что позволяет ей запоминать закономерности в разных последовательностях. В других случаях, это может навредить, но так как последовательности аккордов используют одни и те же правила, ситуация должна измениться к лучшему.

В результате обучения, Stateful модель показала точность 65% на мажор-минор и септаккордах. Дальнейшая манипуляция гиперпараметрами (размер слоев, количество слоев, нормализация, dropout) не привели к улучшению модели.

Тестовый запуск на песне SZA — Kill Bill, показал достаточно хаотичные результаты, из которых тяжело понять изначальную проблему.

Для данного эксперимента используем фреймворк CRFsuite. Эта реализация условных случайных полей (CRF) предназначена для маркировки последовательных данных. Формат данных аналогичен используемому в других инструментах машинного обучения; каждая строка состоит из метки и атрибутов (признаков) элемента, последовательные строки представляют собой последовательность элементов (пустая строка обозначает конец последовательности элементов).

Проводим все операции с данными, которые проводили для обучения LSTM: разбиение на последовательности, паддинг.

Для обучения CRFsuite.CRF необходим специальный формат тренировочных данных. Все последовательности разбиваются на "слова в нашем случае состоящие из множества признаков. Для каждого элемента последовательности зададим все необходимые признаки: тюнинг, номер песни, время, значения хромаграмм для 12 семитонов.

Затем, для каждого элемента добавим признаки предыдущего и следующего элемента, с помощью которых CRF сможет описать более точные переходы между состояниями.

Для первого и последнего элемента последовательности указываем теги «начало» и «конец» строки соответственно.

После 100 итераций обучения модель показывает точность 70% и F1 71%.

Самыми вероятными переходами между аккордами модель посчитала переходы между одинаковыми аккордами. После ручного просмотра результатов

стало очевидно, что модель обучилась неправильным паттернам, а в некоторых случаях выдает длинные последовательности одинаковых аккордов.

Для данного эксперимента используем библиотеку Catboost [10]. Она предоставляет алгоритм градиентного бустинга над решающими деревьями, который может использоваться для задач классификации, регрессии и ранжирования. Библиотека также обладает рядом других особенностей, таких как автоматическое определение оптимального числа деревьев, автоматическое обнаружение переобучения, возможность использования CPU и GPU для обучения и предсказания модели.

Для обучения модели использован класс CatBoostClassifier, предназначенный для работы с задачей классификации. Функция потерь – MultiClass (множество классов), 10% тренировочных данных – валидационная выборка.

В результате 2000 итераций, модель показывает точность 63% на данных включающих септаккорды и 78% на данных без септаккордов.

Анализ результатов показывает, что точность на септаккордах составляет 38%, что очевидно требует улучшения. Также с точностью 50% модель предсказывает класс «без аккорда», что указывает на возможные проблемы с разметкой аккордов.

Вследствие использования модели с песней не из тренировочного или тестового датасета, было выявлено, что получаемые последовательности сильно фрагментированы. В случаях, где модель не уверена в результате, происходят выбросы, что делает использование таких последовательностей практически невозможным. Такие результаты не удивительны, потому что модель изначально не предназначена для работы с последовательными данными.

Влияние данной проблемы можно уменьшить множеством способов, начиная с увеличения размера скользящего окна во время использования модели, возможно применение скрытых марковских моделей или случайных полей для сглаживания последовательности, не исключено использование медианного фильтра, чтобы избавиться от выбросов на маленьких временных отрезках.

Другим решением может быть дополнительная работа с тренировочными данными. Так как изначально в модель Catboost данные поступают целиком, новый признак, отвечающий за номер песни, которой принадлежит строка с данными может позволить модели лучше понимать природу данных.

Важен так же момент, который происходит на разделении данных на

тренировочную и тестовую выборку. Обычно, на этом этапе данные перемешиваются, но так как в этом случае данные не разделены на последовательности, это перемешивает все последовательности друг с другом. Однако отказ от перемешивания данных может вызвать переобучение модели.

Чтобы еще сильнее утвердить последовательную природу данных, попробуем воспользоваться опытом случайных полей и добавить в каждую строку данные о предыдущем и (или) следующем элементе последовательности.

Очевидно, что модель с использованием новых признаков показывает себя гораздо лучше чем модель с использованием классовых весов. Однако несмотря на высокую точность 88%, проблема фрагментированности все еще присутствует. К тому же, несмотря на резкое падение точности после обучения без перемешивания, последовательности получаемые такими моделями кажутся гораздо более читаемыми.

Была также получена модель с точностью 83% на варианте датасета с инверсиями, однако её громоздкий вес не позволяет использовать её в дальнейшем.

Из всех обученных моделей будет использована модель Catboost с точностью 88%.

Для того, чтобы использовать модель для предсказаний, из полученного аудиофайла нужно выделить все признаки, которые использовались при обучении.

После загрузки модели, отправляем ей на вход набор собранных данных. Полученную последовательность аккордов обрабатываем, объединяя смежные отрезки с одинаковыми аккордами в один. Возвращаем пользователю файл с разметкой.

Пайплайн разворачиваем на сервере с помощью FastApi. Это фреймворк для создания лаконичных и довольно быстрых HTTP API-серверов со встроенными валидацией, сериализацией и асинхронностью.

Взаимодействие с API происходит посредством HTTP запросов. Реализуем POST запрос для старта пайплайна с возможностью загрузки файла. После вызова метода predict, пользователь получает response с разметкой аккордов.

ЗАКЛЮЧЕНИЕ

В процессе написания работы был проведен обзор основной теории, научных публикаций необходимых для понимания предметной области, был

проведен анализ существующих решений для распознавания последовательности аккордов аудиосигнала. Проанализировав эффективность существующих методов, выбраны и реализованы лучшие из них. Для моделей рекуррентных нейронных сетей и марковских случайных полей получены показатели выше средних.

Впервые задача распознавания аккордов была решена с помощью моделей решающих деревьев. Обученная модель способна предсказывать последовательности аккордов с высокой точностью, используя только аудиосигнал в качестве входных данных. Для модели было разработано API, что позволяет без труда использовать пайплайн для предсказания аккордов.

Работа апробирована на студенческой научной конференции «Компьютерные науки и информационные технологии» 2023 г.

Данная тема, имеет широчайшее применение в современных реалиях, её изучение открывает возможности в автоматической транскрибции, музыкальном анализе.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 *Набиуллин, Д. А.* Определение жанра музыки с помощью сверточной нейронной сети / Д. А. Набиуллин // *Аллея науки*. — 2018. — Vol. 8. — Pp. 80–83.

2 *Korzeniowski, F.* Feature learning for chord recognition: The deep chroma extractor / F. Korzeniowski, G. Widmer // *CoRR*. — 2016. — Vol. 05.

3 *Muller, M.* Making chroma features more robust to timbre changes / M. Muller, S. Ewert, S. Kreuzer // *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*. — 2009. — Pp. 1877–1880.

4 Kaggle Piano Triads Dataset, [Электронный ресурс]. — URL: https://www.kaggle.com/datasets/davidbroberts/piano-triads-wavset (Дата обращения 29.05.2022). Загл. с экр. Яз. англ.

5 Преобразование с постоянной Q [Электронный ресурс]. — URL: https://wikieto.ru/wiki/Constant-Q_transform (Дата обращения 29.05.2022). Загл. с экр. Яз. англ.

6 *Мюллер, А.* Введение в машинное обучение с помощью Python. / А. Мюллер, С. Гвидо. — Москва: Диалектика, 2017. — 480 с.

7 *Geron, A.* Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow / A. Geron. — Sebastopol: O'Reilly Media, 2019. — 346 pp.

8 Scikit-learn User Guide [Электронный ресурс]. — URL: https://scikit-learn.org/stable/user_guide.html (Дата обращения 29.05.2022). Загл. с экр. Яз. англ.

9 *John Ashley Burgoyne, J. W.* An expert ground truth set for audio chord recognition and music analysis // Proceedings of the 12th International Society for Music Information Retrieval Conference.

10 Catboost [Электронный ресурс]. — URL: https://catboost.ai/en/docs/ (Дата обращения 22.03.2023). Загл. с экр. Яз. англ.