Министерство образования Республики Беларусь
Учреждение образования
Белорусский государственный университет
информатики и радиоэлектроники

УДК 004.032.324:78

Андрадэ
Александр Исмаэлевич

Система распознавания музыкальной транскрипции при помощи методов машинного обучения

ДИССЕРТАЦИЯ

на соискание степени магистра информатики и вычислительной техники

по специальности 1-40 81 02 «Технологии виртуализации и облачных вычислений»

_____

Научный руководитель
Насуро Екатерина Валериевна
к.т.н., доцент

_____

Минск 2019

ОГЛАВЛЕНИЕ

ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ И ТЕРМИНОВ

AUC	Площадь, ограниченная ROC-кривой и осью доли ложных положительных классификаций. Чем выше показатель AUC, тем качественнее классификатор, при этом значение 0,5 демонстрирует непригодность выбранного метода классификации (соответствует случайному гаданию)
CQT	Преобразование, которое трансформирует временные ряды данных в частотную область. Оно связано с преобразованием Фурье и очень тесно связан с комплексным вейвлет-преобразованием Морле
CWT	Непрерывное вейвлет-преобразование
DSMC	Цепочка марковских дискретных состояний
FFT	Быстрое преобразование Фурье (БПФ)
HMM	Скрытая марковская модель
LSTM	Long short-term memory — разновидность архитектуры рекуррентных нейронных сетей, предложенная в 1997 году Сеппом Хохрайтером и Юргеном Шмидхубером
MDP	Марковским процессом принятия решений
MFCC	Мел-кепстральные коэффициенты, представление лог мощности спектра в мел частотной области. Описывают мощность огибающей спектра, которая (огибающая) характеризует модель речевого тракта.
POMDP	Частично наблюдаемые марковские процессы принятия решений.
RNN-RBM	Является энергетической моделью для оценки плотности временных последовательностей, где вектор признаков v(t) на шаге t времени может быть многомерным.
ROC	График, позволяющий оценить качество бинарной классификации, отображает соотношение между долей объектов от общего количества носителей признака, верно классифицированных как несущих признак и долей объектов от общего количества объектов, не несущих признака, ошибочно классифицированных как несущих признак
STFT	Оконное преобразование Фурье — это разновидность преобразования Фурье.
SVM	Метод опорных векторов (support vector machine) — набор алгоритмов обучения с учителем, использующихся для задач классификации и регрессионного анализа.

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Цель работы: состоит в исследовании методов музыкальной транскрипции и реализации метода, основанного на использовании рекуррентной нейронной сети.

Задачи исследования: В соответствии с поставленной целью, можно выделить следующие задачи:

1. Выполнить анализ предметной области.
2. Так как задача не решена в общем виде на сегодняшний день, предложить свой метод для решения задачи транскрипции фортепиано.
3. Провести испытания, получить результаты работы метода, если возможно – улучшить их.
4. Оценить, насколько хорошо метод решает задачу, определить его преимущества и недостатки.
5. Предложить улучшения для предложенного метода, которые, возможно, позволят получить более качественный результат.

Гипотезы:

1. Частотно-временное представление произведения фортепиано имеет достаточно информации (свойств) для точной классификации каждой ноты в определенный момент времени.
2. Временная составляющая информации (свойств) имеет важное значение, поэтому для решения задачи подходящим инструментом послужат рекуррентные сети типа LSTM.
3. Для работы классификатора достаточно обучить одну нейросеть с независимыми классификаторами, число которых равно колву нот фортепиано.
4. Необходимую информацию (свойства) нейросеть способна выделить из предоставляемого частотно-временное представления.

Данная работа имеет практическую *связь* с такими направлениями, как распознавание речи и цифровая обработка сигнала, поэтому она актуальна.

Результаты исследований опубликованы в виде тезисов на следующих научных конференциях:

1. 54-я научно-техническая конференция аспирантов, магистрантов и студентов БГУИР, которая проходила в мае 2019 в Минске. Тема тезисов: «Музыкальная транскрипция при помощи методов машинного обучения» [1 – А.].
2. Пятая международная научно-практическая конференция «Big Data and Advanced Analytics», 14.03.2019 в Минске. Тезисы: «Средство музыкальной транскрипции при помощи методов машинного обучения» [2 – А.].

ВВЕДЕНИЕ

Музыкальная транскрипция – это процесс анализа акустического музыкального сигнала и запись музыкальных параметров звуков, возникающих в нем. Традиционно музыка записывается в виде нотной записи, которая позволяет указать высоту тона, время начала и продолжительность каждого воспроизводимого звука.

Музыкальную транскрипцию также можно представить в репрезентационном смысле как преобразование акустического сигнала в символическое представление. С другой стороны, музыкальная нотация - это скорее инструкция для исполнителя, чем представление музыкальной записи.

Она описывает музыку на языке, понимаемом музыкантом и служит для воспроизводства музыкальных звуков. С этой точки зрения, музыкальная транскрипция может быть рассмотрена как обнаружение «рецепта» или реверс-инжиниринг «исходного кода» музыкального сигнала. Прикладная нотация не обязательно должна быть традиционной музыкальной нотацией, скорее любым символическим представлением, которое дает необходимую информацию для исполнения записи используя доступные музыкальные инструменты. В случае электронного синтезатора, используемого для повторного синтеза, MIDI файл является примером такого музыкального представления записи.

Музыкальная нотация не только позволяет воспроизводить музыкальную запись, но также и модифицировать, поправлять и обрабатывать музыку на более высоком уровне абстракции. Другое важное применение музыкальной транскрипции – это структурированное аудио кодирование. MIDI-подобное представление невероятно компактно, но в то же время оно сохраняет идентифицируемость и характеристики музыкального произведения в значительной степени. Еще одно применение музыкальной транскрипции включает *поиск информации* на основе мелодии фрагмента, *музыкальный анализ* импровизируемой этнической музыки и *интерактивные музыкальные системы,* которые генерируют аккомпанемент поющему или играющему солисту.

Автоматическая транскрипция полифонической музыки является субъектом все возрастающих исследований на протяжении последних 30 лет. Проблема во многом аналогична автоматическому распознаванию речи, но не получила сопоставимых академического или коммерческого применения.

Несмотря на большое количество попыток решить проблему, практически применимая, универсальная система музыкальной транскрипции не существует и по сей день. Существующие системы на порядок хуже

специалистов в точности и гибкости. Самые новые системы, однако, достигли определенной степени точности в транскрибировании полифонической музыки ограниченной сложности. Типичные ограничения целевого сигнала таковы:

– число одновременных нот ограниченно (или фиксировано);

– интерференция ударных и перкуссионных инструментов обычно запрещена;

Как ни странно, даже транскрипция одноголосого произведения является еще нерешенной проблемой.

Цель работы: состоит в разработке метода автоматической транскрипции, который по возможности учитывает опыт и недостатки аналогов для получения лучшего результата распознавания.

С учётом описанных выше недостатков существующих подходов, для достижения поставленной цели перед автором данной работы были поставлены следующие *задачи*:

3. Разработать метод для более точного выделения в спектре звука информации, с целью получения более аккуратного отображения фрагментов звукозаписи в пространство признаков.

4. Выделить новые наиболее информативные признаки, желательно небольшой размерности, например, использовать мел-кепстральные коэффициенты (MFCC) в качестве признаков.

5. Реализовать описанные алгоритмы в виде программного продукта, позволяющего распознавать последовательность нот в поданном на вход звуковом файле.

ГЛАВА 1
ОБЗОР ЛИТЕРАТУРЫ

1.1 Звук

Звук – объективно существующее в природе физическое явление, вызываемое механическими колебаниями какого-либо упругого тела, вследствие чего образуются звуковые волны, воспринимаемые ухом и преобразуемые в нём в нервные импульсы. Звуковыми волнами называются периодически чередующиеся сгущения и разрежения в какой-либо упругой (т.е. звукопроводящей) среде; звуковые волны воспринимаются слуховыми органами человека и животных и при помощи центростремительных нервов передаются в большие полушария головного мозга, где и осознаются как конкретные звуки.

Все звуки вокруг нас распадаются на 2 типа: с определенной высотой (музыкальные звуки) и с неопределённой высотой (шумовые звуки). Музыкальные звуки составляют звуковой фонд музыки, в то время как шумовые звуки применяются лишь эпизодически. Музыкальный звук имеет 4 основных свойства: высота, длительность, громкость, тембр [6].

1.2 Свойства звука

Высота. Высота звука обусловлена частотой колебаний вибратора и находится от неё в прямой зависимости. Частота колебаний находится в обратной зависимости от величины (длины и толщины) звучащего тела и в прямой – от упругости.

Слух человека воспринимает звуки в диапазоне частот от 16 до 20000 Гц, в раннем детстве до 22000 Гц, в старости до 14000-15000Гц. Наиболее точно и ясно человек воспринимает звуки в пределах от 16 Гц до 4200-4500 Гц, этот диапазон и используется в музыке. Зависимость между частотой колебаний и высотой звука – геометрическая прогрессия. При увеличении частоты на 110 Гц (это приблизительно соответствует укорачиванию струны в два раза) от А (110Гц) образуются интервалы: ч.8, ч.5, ч.4, б. 3, м.3, м.3, несколько б.2, несколько м.2. Дальше образуются интервалы меньше полутона. Этот звуковой ряд соответствует натуральному ряду чисел и называется натуральным

звукорядом. Его можно получить при делении струны на 2, 3, 4, 5, 6 и т.д. частей, чем пользуются при исполнении на струнных инструментах флажолетов. Эталон высоты звука – 440 Гц (ля первой октавы).

Акустическая единица измерения звуковысотных расстояний – цент = 1/100 темперированного полутона. Порог различения изменения высоты звука в среднем регистре – 5 центов.

Длительность. Длительность звука – выраженное в ритмических единицах время, в течение которого совершаются колебательные движения вибратора. Прямая зависимость. Длительность музыкального звука колеблется от 0,015-0,02 с до нескольких минут (педальные звуки органа). В тактовой нотации (с 17 в.) ноты указывают лишь относительную длительность звука, реальное значение которой зависит от темпа.

Громкость. Громкость звука – отражение в восприятии силы звука, обусловленной амплитудой колебаний. Применяемые в музыкальной практике обозначения динамических оттенков показывают не абсолютные значения громкости звука, а соотношения между их градациями.

Колебания бывают 2 видов: затухающие (т.е. с постепенно уменьшающейся за счёт сопротивления воздуха и внутреннего торможения амплитудой – рояль, арфа, струнно-щипковые) и незатухающие (с постоянной или произвольно меняющейся амплитудой – орган, скрипка при игре смычком). При затухающих колебаниях громкость звука постепенно уменьшается до полного затихания (высота остаётся практически неизменной). При незатухающих колебаниях громкость можно варьировать в зависимости от художественных целей.

Интенсивность (сила) звука – отношение падающей на поверхность звуковой мощности к площади этой поверхности, измеряется в Вт/м2. При росте силы звука в геометрической прогрессии громкость возрастает лишь в арифметической.

1.3 Некоторые понятия из теории музыки

Человек воспринимает звуки с частотами f_0 и $2f_0$ (до 5000 Гц) как очень похожие и тесно связанные друг с другом. Расстояние между такими звуками называется октавой. Даже некоторые животные – например, обезьяны и кошки, – воспринимают звуки, отличающиеся на октаву, как похожие.

Звукоряд делится на октавы на основе октавного сходства его звуков и отражающей это сходство повторности их названий. В свою очередь, каждая

октава имеет своё название: субконтроктава, контроктава, большая октава, малая октава и октавы с первой по пятую. Началом октавы принято считать звук ступени до.

Темперированным называется строй, который делит каждую октаву звукоряда на равные части. С начала XVIII века в европейской музыке принята двенадцатизвуковая (двенадцатиступенная) темперация, делящая октаву на 12 равных друг другу частей, называемых полутонами.

Полутон является наименьшим расстоянием по высоте, возможным в двенадцатизвуковом темперированном строе. Он образуется между звуками любых двух соседних ступеней звукоряда.

Расстояние, образованное двумя полутонами, называется целым тоном. Расстояние между двумя соседними основными ступенями звукоряда (соответствующими белым клавишам фортепиано) может быть равно полутону (например, ми–фа) или целому тону (например, фа–соль).

Частоту каждой ступени звукоряда можно вычислить по формуле

$$f_i = f_0 \cdot 2^{i/12}, \tag{1.1}$$

где $f0$ – частота настройки музыкальных инструментов. В 1939 году на международной конференции в Лондоне был принят стандарт для частоты настройки $f0 = 440$ Гц. Эту частоту фиксируют для звука ля первой октавы.

Клавиатура фортепиано охватывает 88 ступеней: от ля субконтроктавы до ступени до пятой октавы. Частота, соответствующая k-й слева клавише фортепиано (отсчитывается с нуля), может быть вычислена по формуле

$$f_k = 27.5 \cdot 2^{k/12}, \tag{1.2}$$

Широко используемый в настоящее время стандарт MIDI (Musical Instrument Digital Interface), задающий формат обмена данными между электронными музыкальными инструментами, определяет 128 возможных значений для частоты звука. Частота, соответствующая ступени с номером k, $0 \leq k \leq 127$, может быть получена по формуле.

$$f_k = 2^{\frac{k-69}{12}} \cdot 440, \tag{1.3}$$

И наоборот, номер ступени может быть получен из частоты по формуле

$$k = 69 + round(12\log_2(\frac{f}{440})), \tag{1.4}$$

Приведенные выше формулы справедливы для стандартного значения частоты настройки f_0 = 440 Гц. В рамках стандарта MIDI звук ля первой октавы соответствует 69-й ступени [7].

1.4 Цифровой звук

Звуковой сигнал $x(t)$ может быть представлен в цифровом виде при помощи операций дискретизации и квантования. Для этого с некоторой частотой ν раз в секунду измеряется амплитуда функции $x(t)$ (дискретизация), после чего каждое полученное значение $x(t_i)$ заменяется на ближайшее из заданного множества X_Q возможных значений амплитуды (квантование). Как правило, это множество содержит 28, 216 или 224 элементов, чтобы каждое значение можно было представить целым числом байт. Частота ν часто выбирается равной 44100 Гц (по историческим причинам). При этом ν называют частотой дискретизации, а значения $x_Q(t_i)$ – отсчётами исходного сигнала $x(t)$). В соответствии с классической теоремой Котельникова, если спектр сигнала $x(t)$ ограничен сверху частотой $\nu/2$ (т.е. ak = 0 для $\frac{\mathrm{k}}{\tau} > \nu/2$), то исходный сигнал может быть восстановлен однозначно и без потерь по измеренным значениям $x(t_i)$. При квантовании эти значения заменяются на $x_Q(t_i)$, поэтому исходный сигнал может быть восстановлен из оцифрованного только с некоторой ошибкой, которая тем меньше, чем больше возможных значений амплитуды использовалось при квантовании. Для большинства звукозаписей эта ошибка незаметна на слух. Отметим ещё раз, что спектр любых оцифрованных звуковых сигналов ограничен [7].

Частотный интервал (frequency bin) - это отрезок $[f_l,f_h]$ оси частот, который «собирает» амплитуду или энергию из небольшого диапазона частот, что часто является результатом анализа Фурье. Из-за дискретизации данных (возможно, из-за выборки), как правило, невозможно назначить точную амплитуду каждой частоте на реальной оси. Частотный интервал может быть получен, например, из частоты дискретизации и частотного разрешения преобразования Фурье. Однако часть вычисленной амплитуды может быть отнесена к частотам фактического сигнала, которые не содержатся в диапазоне бина. Это явление называется утечкой или смазыванием и зависит от инструментов, используемых для получения этой амплитуды.

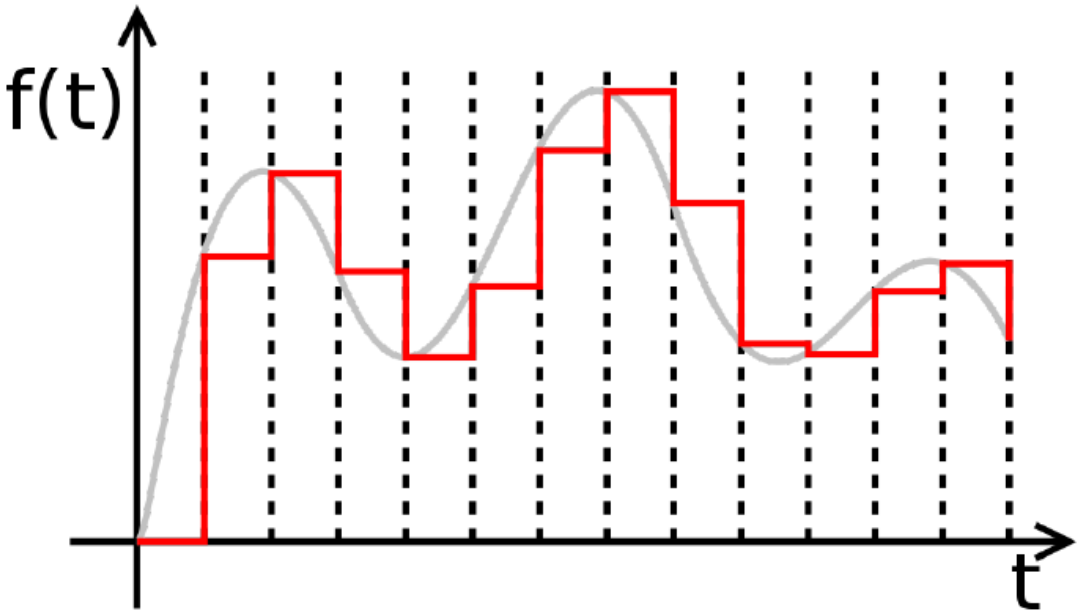

Рисунок 1.1 – Дискретизация и квантование

1.5 Свойства музыкальных звукозаписей

Формат файла определяет структуру и особенности представления звуковых данных при хранении на компьютере. Для устранения избыточности данных используются аудиокодеки, при помощи которых производится сжатие аудиоданных. Выделяют три группы звуковых форматов аудиофайлов:

- форматы без сжатия (WAV, AIFF);
- форматы со сжатием без потерь (APE, FLAC);
- форматы с применением сжатия с потерями (mp3, ogg).

Название формата	Квантование, бит	Частота дискретизации, кГц	Число каналов	Скорость потока данных с диска, кбит/с	Степень и тип сжатия
CD	16	44,1	2	1411,2	1:1 без потерь
Dolby Digital (AC3)	16-24	48	6	до 640	~12:1 с потерями
DTS	20-24	48; 96	до 8	до 1536	3:1 с потерями
DVD-Audio	16; 20; 24	44,1; 48; 88,2; 96	6	6912	1:1 без потерь
DVD-Audio	16; 20; 24	176,4; 192	2	4608	1:1 без потерь
MP3	16-24	до 48	2	до 320	~11:1 с потерями
AAC	16-24	до 96	до 48	до 512	с потерями
AAC+ (SBR)	16-24	до 48	2	до 320	с потерями
Ogg Vorbis	до 32	до 192	до 255	до 500	с потерями
WMA	до 24	до 96	до 2	до 768	2:1, есть версия без потерь

Рисунок 1.2 – Некоторые виды цифровых аудиоформатов в сравнении

Музыкальные звукозаписи в целом обладают рядом свойств, которые

нужно учитывать при транскрибировании:

1. Наличие гармоник у музыкальных инструментов с ясно выраженной высотой звучания. В звучании таких инструментов можно выделить отдельную ноту. При этом наряду с частотой, соответствующей этой основной ноте, звучат другие частоты. Их звучание менее выражено, но они могут соответствовать другим ступеням музыкальной системы. Математически это означает, что если k_0 таково, что a_{k_0} – наибольшая по абсолютному значению компонента спектра звучащей ноты, то существует по меньшей мере одно значение $k > k_0$ такое, что a_k существенно отлична от 0. Соотношения между парами (k_0, k) для разных k и разных k_0 (соответствующих разным нотам) во многом задают тембр музыкального инструмента.

2. Наличие периодичностей (повторений). Одна и та же музыкальная фраза, последовательность аккордов и даже целый фрагмент композиции могут повториться в точности или с небольшими изменениями [7].

1.6 Существующие подходы и их слабые стороны

Мурер [1977] впервые представил ограниченную систему для транскрипции дуэта. С тех пор ряд акустических моделей для полифонической транскрипции были представлены как в частотной области [Rossi et al., 1997, Sterian, 1999, Dixon, 2000], так и во временной [Bello et al., 2002].

Однако эти методы основаны на анализе, который предполагает, что сигнал будет иметь определенную структуру. Музыкальный тон создается с периодичностью на определенной основной частоте в звуковом сигнале.

Предположение, что высота ноты возникает из гармонических составляющих сильно основано на музыкальной акустике, но для транскрипции это необязательно. Во многих направлениях, таких как автоматическое распознавание речи, классификаторы для определенных событий построены с использованием минимальных предварительных знаний об используемых (значительных) признаках. Marolt [2004] представил классификационный подход с использованием нейронных сетей, но блок фильтров адаптивных осцилляторов был необходимых для уменьшения ошибочных вставок нот. Байесовские модели также были предложены для транскрипции музыки, однако, такое их трактование слишком основано на физическом представлении модели генерации музыкального звука [1].

К сожалению, взаимодействие гармонических рядов, происходящее в полифонической музыке значительно запутывает автоматическую

транскрипцию.

Poliner, Ellis (2006):

– метод опорных векторов и обучение на спектральных свойствах для классификации нот во фрейме;

– выход классификатора пропускается через скрытые марковские модели;

– система используется для транскрипции как синтезированных, так и живых записей фортепиано [2].

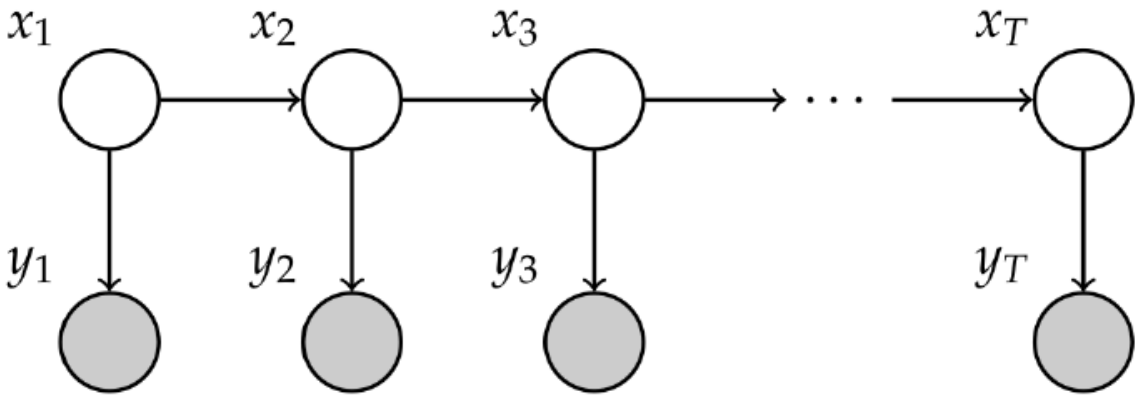

Рисунок 1.3 – Пост-процессинг скрытыми марковскими моделями

Ryyn ̈anen, Klapuri (2005):

– первичная обработка — multiple-F0 estimator;

– HMM для нот, модель тишины и музыкологическая модель;

– музыкологическая модель отвечает за поиск переходов между HMM и моделью тишины.

События нажатия нот описываются скрытой марковской моделью. Модель использует три акустические характеристики с оценкой множественной основной частоты (F0) для вычисления вероятности различных нот и выполняет временную сегментацию нот. Переходы между нотами контролируются с помощью музыкальной модели с оценкой музыкального ключа и bigram моделей. Окончательная транскрипция получается путем поиска нескольких путей через модели нот [3].

Система Klapuri вычисляет множественные фундаментальные частоты из спектральных пиков, используя вычислительную модель слуховой периферии человека. Затем дискретные скрытые марковские модели (HMM) итеративно применяются для извлечения последовательности мелодий из фундаментальных оценок частоты [3]. На рисунке 1.4 приведена блок-диаграмма метода:

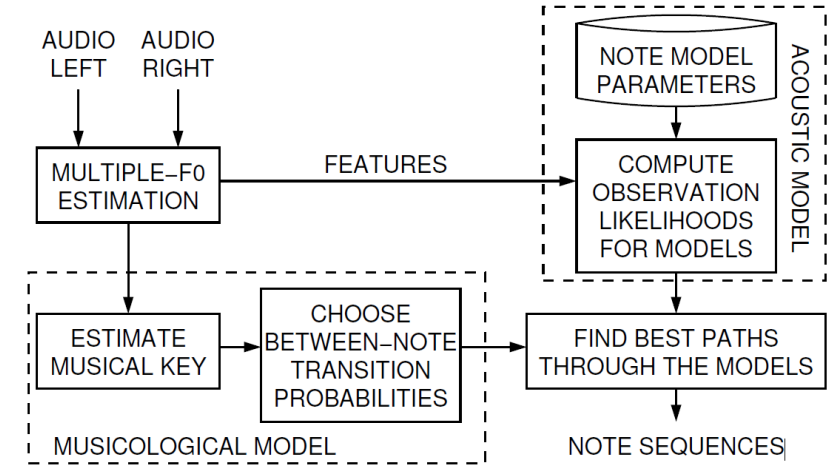

Рисунок 1.4 – Блок-диаграмма метода Ryyn ̈anen, Klapuri (2005)

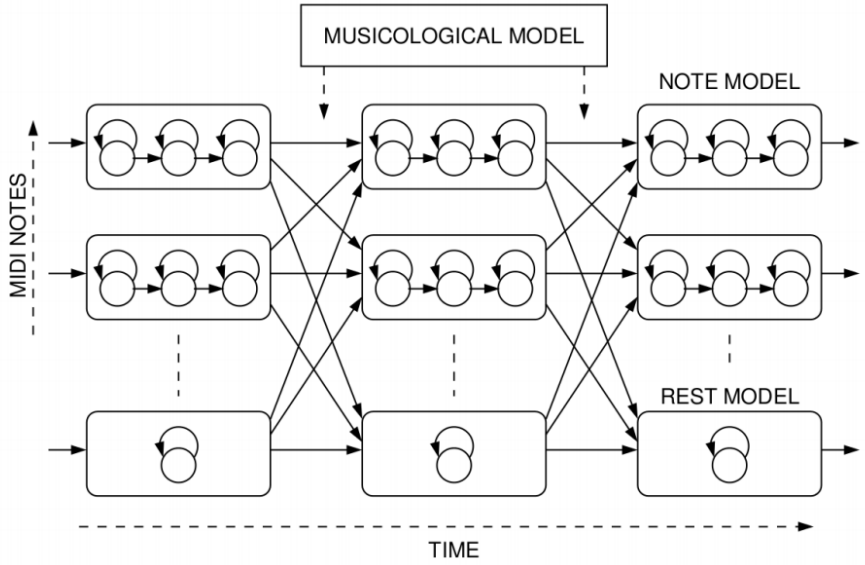

Рисунок 1.5 – Сеть модели ноты и модели тишины

Marolt (2004):

– используются нейросети и их обобщения;

– вводятся «адаптивные осцилляторы» для определения частот звуков;

– с помощью особой синхронизации осцилляторов удается учитывать обертоны в музыке;

– объединяя такие осцилляторы в сети, получим систему для отслеживания целых групп обертонов, т.е. фактически получаем необходимую транскрипцию.

На рисунке 1.5 - модель отслеживания частот, нейронная сеть для учета временных задержек. Система также содержит детектор появления ноты, модуль для детектирования повторяющихся нот и простой алгоритм для оценки длины и громкости [4].

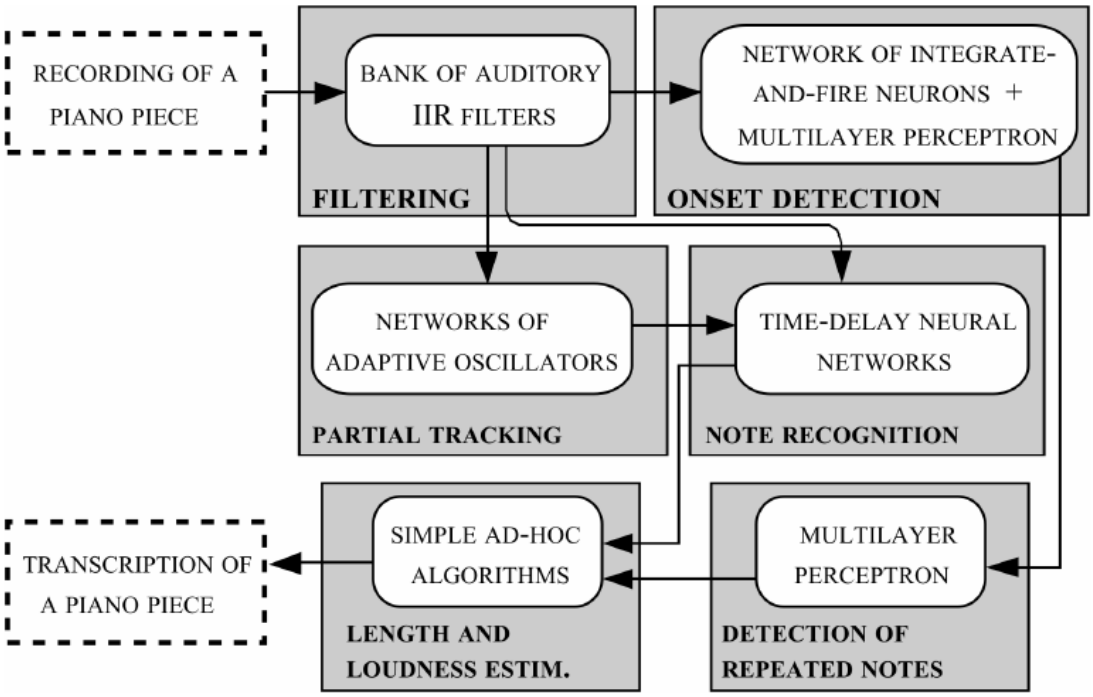

Рисунок 1.6 – Структура системы Marolt (2004)

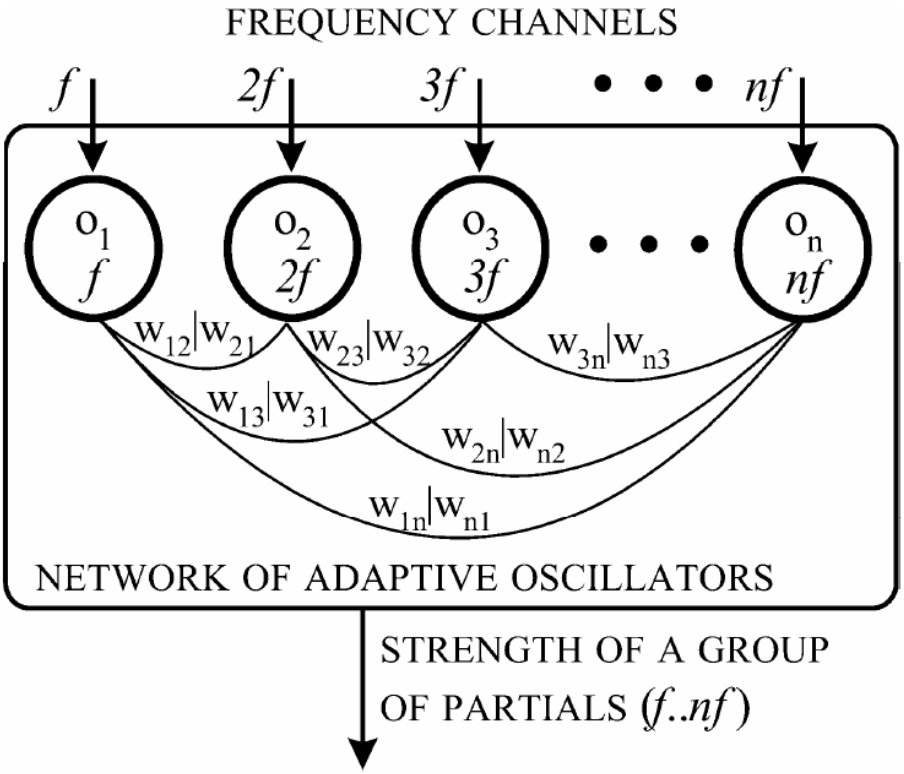

Рисунок 1.7 – Сеть адаптивных осцилляторов

Сеть осцилляторов может доходить до десяти взаимосвязанных осцилляторов. Их начальные частоты устанавливаются как целые кратные частоте первого осциллятора. Система использует 88 сетей осцилляторов для отслеживания групп, соответствующих всем 88 тонам фортепиано (A0-C8).

Начальная частота первого осциллятора в каждой сети установлена на высоту одного из 88 тонов фортепиано (см рисунок 1.7).

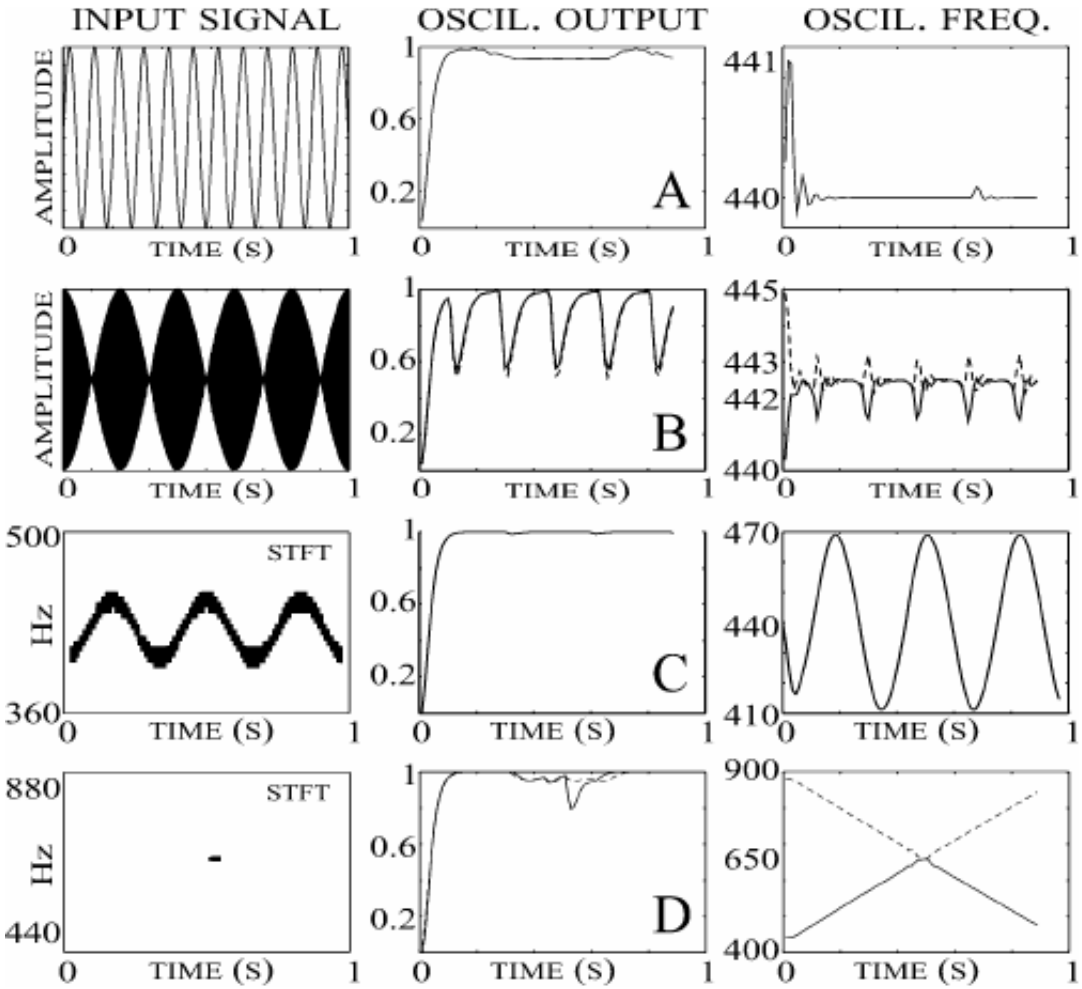

Рисунок 1.8 – Отслеживание частот адаптивными осцилляторами

Zalani, Mittal (2014):

- RNN-RBM для обучения признаков;
- STFT тоже учитывается как признаки;
- 88 SVM-классификаторов One-vs-All;
- пост-процессинг скрытыми марковскими моделями.

В этом подходе обучают 88 бинарных классификаторов, которые служат для транскрибирования нот в полифонической записи (см. рисунок 1.9). Каждый классификатор определяет присутствие одной ноты в записи в каждый отдельный отсчет времени. Обучение без учителя RNN-RBM (Рекурсивные нейронные сети и ограниченная машина Больцмана). Convolutional Deep Belief Network (CDBN) применены. Классификация методом опорных векторов (SVM) «один против всех» one-vs-all. HMM сглаживание для улучшения результатов [5].

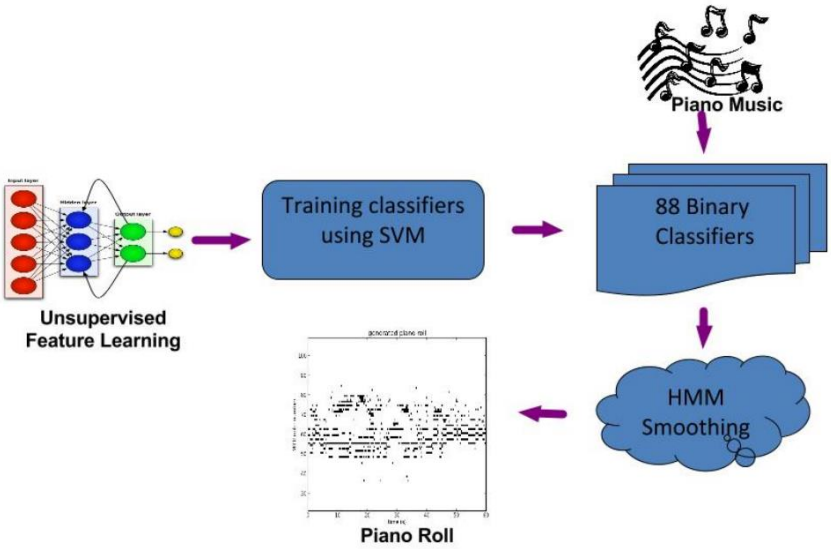

Рисунок 1.9 – Подход Zalani, Mittal (2014)

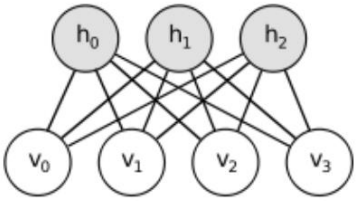

Рисунок 1.10 – Ограниченная машина Больцмана

v_i представляет видимые блоки, h_i – скрытые (см рисунок 1.10).

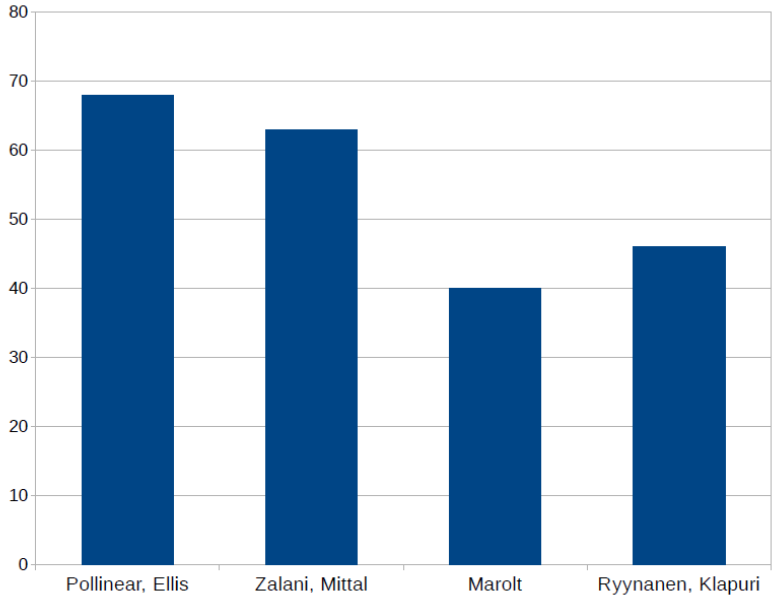

Рисунок 1.11 – Сравнение точности распознавания аналогов

Сравнение точности распознавания аналогов представлено на рисунке 1.11.

ГЛАВА 2
РАСПОЗНАВАНИЕ НОТ С ПОМОЩЬЮ МАШИННОГО ОБУЧЕНИЯ

2.1 Формализация задачи

Теперь, с использованием введённых понятий и обозначений, можно дать формальную постановку основной задачи, решаемой в данной работе: пусть заданы звуковой сигнал $x(t)$, $t \in [\mathrm{t}_{\mathrm{start}}, \mathrm{t}_{\mathrm{end}}]$ и множество возможных названий аккордов Y. Необходимо для каждого момента времени $t \in [\mathrm{t}_{\mathrm{start}}, \mathrm{t}_{\mathrm{end}}]$ указать аккорд $y \in Y$, звучащий в этот момент. Эту общую задачу можно разделить на отдельные шаги, которые соответствуют конкретным задачам, перечисленным во введении [7].

2.1.1 Частотно-временное представление

Представление звука в виде последовательности отсчётов амплитуды не является удобным для обработки: неясно, как сопоставить аккорду последовательность отсчётов и наоборот. Поэтому естественным первым шагом является часто используемый при обработке звука переход к частотно-временному представлению звукозаписи или получение её спектрограммы $CN{\times}M$. Она представляет из себя матрицу, каждая из N строк которой соответствует определённой частоте, а каждый из M столбцов представляет из себя спектр фрагмента звукозаписи в пределах короткого промежутка. Элементами этой матрицы являются значения интенсивности данной частоты в пределах данного промежутка времени.

Частотно-временное представление может быть получено с помощью разных техник Short-time Fourier transform (STFT), Continuous wavelet transform (CWT) или Constant-Q transform (CQT).

2.1.2 Классификация

Имеем множество всех возможных векторов-столбцов спектрограммы.

Для каждого вектора из данной последовательности необходимо указать множество нот, соответствующих этому вектору. Принимая во внимание известные из равномерно темперированного строя частоты нот, можно по спектру звука в данном фрагменте делать предположения относительно звучащих нот и аккорда. Возможно использование не только текущего, но и предыдущих векторов (а также последующих, если от алгоритма не требуется выдавать результат в реальном времени). Также возможно использование результатов классификации других векторов. На этом этапе главными вопросами являются выбор метода классификации и отыскание такого набора преобразований множества, который позволит упростить классификацию с использованием выбранного метода. Результатом преобразования будет последовательность векторов признаков, отличная от исходной последовательности векторов-столбцов спектрограммы.

2.1.3 Предварительная обработка

Предварительная обработка может содержать следующие шаги:

– преобразование стерофонических записей в монофонические;

– возможно понижение частоты дискретизации с 44100 Гц до 11025 Гц, при этом теряются частоты выше 5.5 кГц, что не является критичным, так как в музыке большая часть информации находится на частотах до 5 кГц;

– преобразование сигнала из временного представления в частотно-временное;

2.1.4 Спектрограмма

Переход к частотно-временному представлению звукозаписи в виде спектрограммы является ключевым, поскольку даёт возможность работать с отдельными частотными компонентами звука. Для этого звукозапись делится на короткие, возможно, пересекающиеся фрагменты, на каждом из которых вычисляется спектр звука.

Для преобразования звукозаписи к частотно-временному представлению могут быть применены следующие алгоритмы, описанные ниже.

STFT (Short-time Fourier transform), получают спектрограмму звукозаписи. Это позволяет работать с отдельными частотными компонентами

звука. Ось x спектрограммы – время, y – частота, цвет – интенсивность определенной частоты в данный момент времени. Для получения спектрограммы сигнал разделяется на временные сегменты с перекрытием, выполняется на каждом сегменте FFT и далее совмещаются все FFT сегментов на одном графике. Недостаток STFT в сложности нахождения компромисса между временным и частотным разрешениями. Вместе с тем, при использовании алгоритмов быстрого преобразования Фурье невозможно произвольным образом выбирать частоты его компонент. Это создает неудобства при дальнейшей обработке, поскольку невозможно точно определить количество звуковой энергии, приходящейся на частоты, соответствующие ступеням звукоряда. При фиксированном размере окна FFT, информация, используемая для расчета FFT ограничена. Если выбрать большое окно, то получим очень хорошее частотное разрешение, но это приведет к перекрытию нот. С другой стороны, выбирая малое окно, достигаем хорошего разделения нот, но частотное представление нот хуже, что, возможно, уменьшит способность разделять ноты.

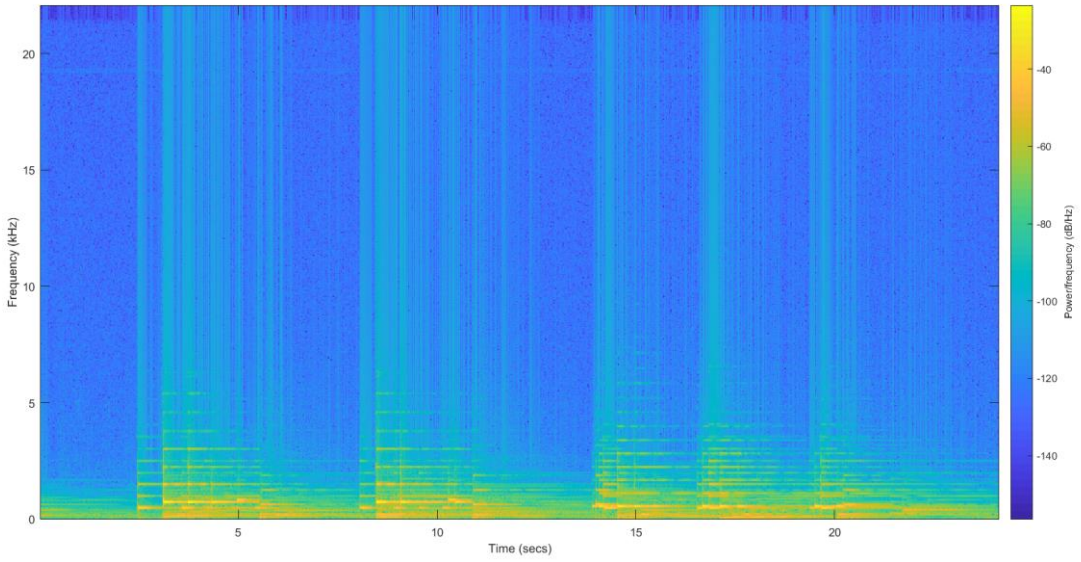

Рисунок 2.1 – STFT участка звукозаписи

В преобразовании постоянного качества (constant-Q преобразование) в отличие от преобразования Фурье, размер анализируемого фрагмента и размер оконной функции зависят от номера соответствующей частотной компоненты. Достоинством этого преобразования является легкость дальнейшей работы со спектром, поскольку его компоненты напрямую соответствуют ступеням звукоряда. Недостатками являются большая сложность вычислений и зависимость от правильного определения частоты настройки.

Гребёнка фильтров (filter bank). В цифровой обработке сигналов любое

преобразование сигнала называют фильтром. В данном случае под фильтром понимается полосовой фильтр – преобразование, сохраняющее в звуке только частоты, находящиеся в некоторой полосе частот. Быстрое преобразование Фурье эквивалентно вполне определенной гребёнке достаточно грубых фильтров. Вместо них можно использовать любые другие фильтры, у каждого из которых центр полосы пропускания соответствует частоте одной из ступеней звукоряда, а ширина полосы пропускания достаточно мала, чтобы не охватывать частоты соседних ступеней. Эти фильтры можно подобрать так, чтобы они были менее грубыми, то есть более точно определяли количество звуковой энергии, приходящейся на их полосы пропускания. Недостатком данного метода является большая вычислительная сложность в сравнении с алгоритмом быстрого преобразования Фурье.

Вейвлет-преобразование Вейвлет-преобразование (WT) – еще одна техника для перевода сигнала из временного представления в частотно-временное. WT является альтернативой STFT и позволяет преодолеть проблемы связанные с неопределенностью частотного и временного разрешений. STFT представляет одинаковое временное разрешение для всех частот, в то время как CWT предоставляет высокое временное разрешение и низкое частотное для высоких частот и высокое частотное разрешение и низкое временное для низких частот. Человеческое ухо демонстрирует аналогичные частотно-временные характеристики разрешения.

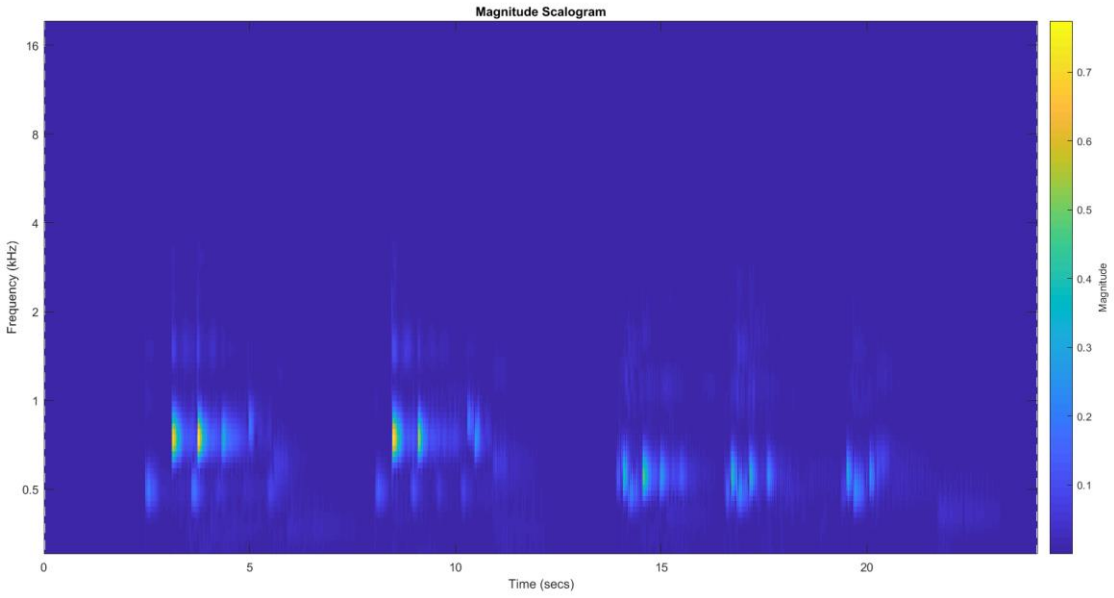

Рисунок 2.2 – CWT участка звукозаписи

2.1.5 Векторы признаков

В качестве признаков могут быть использованы:

- коэффициенты MFCC;
- коэффициенты вейвлет-преобразования;
- столбцы спектрограммы;
- хроматические вектора - векторы признаков, полученные путём объединения спектральной информации по всем октавам, каждый хроматический вектор имеет всего 12 компонент;

2.2 Машинное обучение

Нынешний потенциал для более точной музыкальной транскрипции является результатом прогресса, достигнутого как в области цифровой обработки сигналов, так и машинного обучения. Машинное обучение является областью информатики, которая включает в себя обучение на данных и использования этой обученной системы для прогнозирования новых данных. Специалисты по машинному обучению разрабатывают модели для решения проблем с использованием машинного обучения. Машинное обучение состоит из подготовки данных, а также обучения и тестирования разработанной модели на данных при настройке параметров для получения оптимальных результатов. Многие успешные продукты включают машинное обучение, в случаях, когда явное программирование правил контекстуально неосуществимо.

В работе используется определенный подход к обучению – обучение с учителем. При обучении с учителем на вход сети подаются пары коррелирующих входных и выходных данных, которые система будет учиться приближать. Система тренируется на этих парах, пока не сходится до уровеня ошибки, который является удовлетворительным. Затем мы можем проверить систему на новых данных, чтобы определить, насколько хорошо сеть может обобщать новые данные [8].

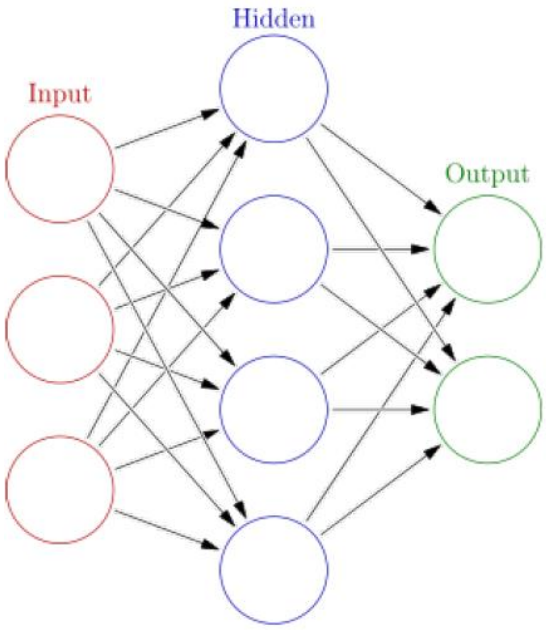

Рисунок 2.3 – Пример модели нейронной сети

2.2.1 Искусственные нейронные сети, функция активации

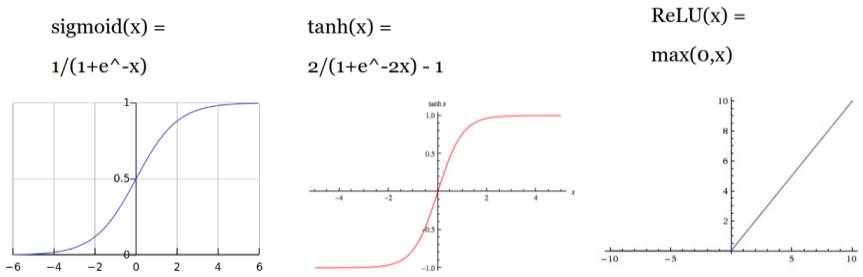

Рисунок 2.4 – Популярные функции активации

Функции активации применяются после выполнения умножения входов на веса каждого полносвязного слоя. Они нормализуют взвешенную сумму каждого значение в результирующем векторе в заданном диапазоне. Рисунок 2.4 показывает некоторые популярные функции активации. При проведении испытаний наилучшие результаты показали ReLU и sigmoid, что будет объяснено в следующих разделах [9].

2.2.2 Предсказания

В самой простой задаче классификации нейронная сеть попытается

определить по свойствам, относится ли экземпляр к определённому классу или нет. Оценка сети, обычно находится в диапазоне от 0 до 1, и мы можем просто округлить прогноз до 0 или 1. Тем не менее, при мультиклассовой классификации, обычно используется softmax слой, который принимает каждую выходную вероятность, преобразует их в экспоненты, и следует, нормализуя каждую вероятность так, чтобы их сумма давала единицу.

Для автоматической транскрипции музыки мы пытаемся определить, какие ноты активированы в группе кадров спектрограммы, зная, что более одной ноты может быть сыгранно одновременно. В контексте нейронных сетей это обозначается как multi-label (потому что более чем одна нота может быть активной в любой момент времени). При multi-label классификации использование softmax функции активации не целесообразно, потому что в этом случае не просто пытаемся выбрать наиболее вероятный единственный класс. Вместо этого пытаемся создать единый порог, чтобы определить, активна или нет каждая нота на основе оценки выходного нейрона, соответствующего этой ноте. Мы можем определить этот порог через сравнение предсказаний вероятностей нот сетью на тренировочных данных с заранее известными правильными классами. Порог это одно число, которое позволяет предсказаниям соответствовать истине как можно ближе.

2.2.3 Рекуррентные нейронные сети Long short-term memory

Рекуррентные нейронные сети - это современный алгоритм для последовательных данных и, среди прочего, используемый Apple Siri и Googles Voice Search. Это связано с тем, что это первый алгоритм, который запоминает свой вход при помощи внутренней памяти, что делает его идеально подходящим для проблем машинного обучения, которые связаны с последовательными данными.

Рекуррентные нейронные сети (RNN) представляют собой мощный и надежный тип нейронных сетей и в настоящее время принадлежат к наиболее перспективным алгоритмам, поскольку они единственные, у которых есть внутренняя память.

Из-за их внутренней памяти, RNN способны помнить важные вещи о том, что они получили, что позволяет им быть очень точными в прогнозировании того, что будет дальше.

Именно по этой причине они являются предпочтительным алгоритмом для последовательных данных, таких как временные ряды, речь, текст,

финансовые данные, аудио, видео, погода и многое другое, поскольку они могут формировать гораздо более глубокое понимание последовательности и ее контекста по сравнению с другими алгоритмы.

Последовательные данные - это просто упорядоченные данные, где связанные вещи следуют друг за другом. Примерами являются финансовые данные или последовательность ДНК. Самый популярный тип последовательных данных - это, пожалуй, данные временных рядов, которые представляют собой лишь ряд точек данных, которые перечислены во временном порядке [10].

В RNN информация циклически проходит через петлю. Когда нейрон сети принимает решение, он принимает во внимание текущий ввод, а также то, что он узнал из ранее полученных им входов.

Два изображения ниже иллюстрируют разницу в потоке информации между RNN и нейронной сетью с обратной связью.

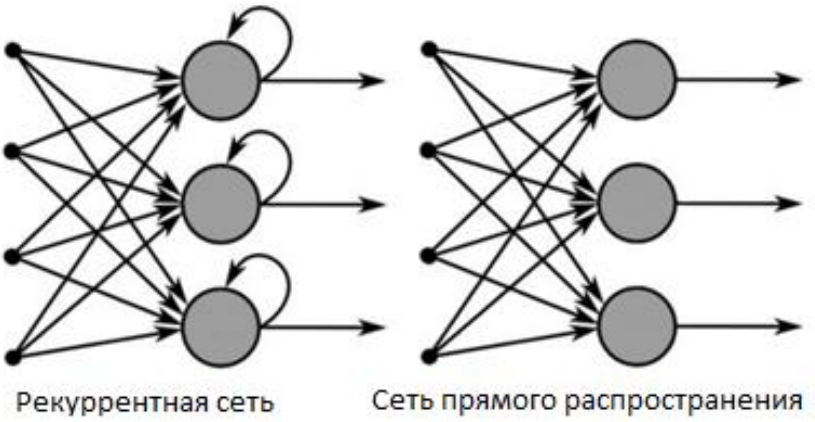

Рисунок 2.6 – Сравнение видов сетей

Обычная RNN имеет кратковременную память. В сочетании с LSTM сеть приобретает также долгосрочную память.

2.2.4 Марковские модели

Перед обсуждением скрытых марковских моделей необходимо рассмотреть более широкое понятие марковской модели. Марковская модель представляет собой стохастическую модель состояния, включающую случайные переходы между состояниями, где вероятность перехода зависит только от текущего состояния, а не от любого из предыдущих состояний. Говорят, что модель обладает марковским свойством и «без памяти». Модели

случайного блуждания - еще один знакомый пример марковской модели [21].

Марковские модели можно классифицировать на четыре широких класса моделей в зависимости от автономии системы и можно ли наблюдать все или часть информации о системе в каждом состоянии. Страница «Модель Маркова» в Википедии дает полезную матрицу, которая описывает эти различия, которые будут повторяться здесь:

Таблица 2.1 – Классы марковских моделей

	Полностью наблюдаемые	Частично наблюдаемые
автономные	Цепь маркова	Скрытая марковская модель
контролируемые	Марковский процесс принятия решений	Частично наблюдаемый Марковский процесс принятия решений

Простейшая модель, Марковская цепь, является автономной и полностью наблюдаемой. Она не может быть изменена действиями «агента», как в контролируемых процессах, и вся информация доступна из модели в любом состоянии. Хорошим примером Марковской цепи является алгоритм Марковской цепи Монте-Карло (MCMC), который в значительной степени используется в вычислительном байесовском выводе.

Если модель все еще полностью автономна, но лишь частично наблюдаема, то она известна как скрытая марковская модель. В такой модели существуют скрытые состояния (и переходы вероятностей между ними), но они не наблюдаются непосредственно и вместо этого влияют на «наблюдения». Важным моментом является то, что, хотя латентные состояния обладают Марковским свойством, для состояний наблюдения не требуется. Наиболее распространенное использование HMM за пределами количественного финансирования - в области распознавания речи.

После того, как система будет «контролироваться» агентом (агентами), тогда такие процессы подпадают под рубрику «Обучение с подкреплением», которое часто считается третьим «столпом» машинного обучения, а также контролируемым обучением и неконтролируемым обучением. Если система полностью наблюдаема, но контролируется, то модель называется Марковским процессом принятия решений (MDP). Связанный с ним метод известен как Q-Learning, который используется для оптимизации политики выбора действий для агента в рамках модели процесса принятия решения по методу Маркова. В 2015 году компания Google DeepMind впервые использовала сети с лубоким подкреплением или Deep Q сети для создания оптимального агента для

воспроизведения видеоигр Atari 2600 исключительно из экранного буфера.

Если система контролируется и только частичнонаблюдаема, то такие модели обучения с подкреплением называются частично наблюдаемыми марковскими процессами принятия решений (POMDP). Методы решения высокоразмерных POMDP являются предметом многих текущих научных исследований. Некоммерческая команда OpenAI проводит значительное время, рассматривая такие проблемы, и выпустила инструментарий с открытым исходным кодом или «спортзал», чтобы обеспечить прямое тестирование новых агентов RL, известных как OpenAI Gym.

К сожалению, обучение с подкреплением, наряду с MDP и POMDP, не входит в рамки этой статьи. Однако они будут предметом более поздних статей, особенно по мере дальнейшего развития серии статей о глубоком обучении.

Заметим, что в этой статье марковские процессы непрерывного времени не рассматриваются. В количественной торговле единица времени часто предоставляется через отсчеты или бары данных исторических активов. Однако, если целью являются производные цены, тогда будет использоваться механизм стохастического исчисления непрерывного времени [24].

2.2.5 Математическая спецификация модели Маркова

Этот раздел, а также то, что в Математической спецификации модели скрытых марковских течений будем строго следовать обозначению и спецификации модели Murphy (2012).

В количественном финансировании анализ временных рядов часто представляет большой интерес. Такой временной ряд обычно состоит из последовательности T дискретных наблюдений X1, ..., XT. Важное предположение о моделях Марковской цепи заключается в том, что в любое время t наблюдение Xt захватывает всю необходимую информацию, необходимую для прогнозирования будущих состояний. Это предположение будет использовано в следующей спецификации.

Формирование Марковской цепи в вероятностную структуру позволяет использовать совместную функцию плотности вероятности наблюдения за наблюдениями как:

$$p(X_{1:T}) = p(X_1)\,p(X_2 \mid X_1)\,p(X_3 \mid X_2)\ldots = p(X_1)\prod_{t=2}^{T} p(X_t \mid X_{t-1}) \qquad (2.1)$$

Это указывает на то, что вероятность наблюдения последовательностей наблюдений определяется вероятностью первоначального наблюдения, умноженной на T-1 раз на условную вероятность наблюдения последующего наблюдения, учитывая предыдущее наблюдение. В этой статье предполагается, что последний член, известный как переходная функция, p (Xt|Xt-1) сама будет независима от времени.

Кроме того, поскольку модели рыночного режима, рассмотренные в этой серии статей, будут состоять из небольшого дискретного числа режимов (или «состояний»), скажем K, тип рассматриваемой модели известен как цепочка марковских дискретных состояний (DSMC).

Таким образом, если есть K отдельных возможных состояний или режимов для модели, которая должна находиться в любое время t, то функция перехода может быть записана как матрица перехода, которая описывает вероятность перехода из состояния j в состояние i на любом временном шаге т. Математически элементы матрицы перехода A задаются следующим образом:

$$A_{ij} = p(X_t = j \mid X_{t-1} = i) \qquad (2.2)$$

В качестве примера можно рассмотреть простую модель цепочки Маркова с двумя состояниями. Следующая диаграмма представляет пронумерованные состояния в виде кругов, а дуги представляют вероятность перехода из состояния в состояние:

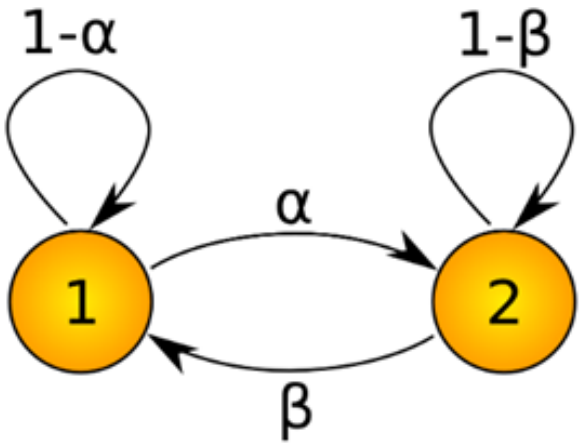

Рисунок 2.11 – Цепь Маркова с двумя состояниями

Обратите внимание, что вероятности суммируются до единицы для каждого состояния, т. Е. A + (1-α) = 1. Матрица перехода A для этой системы представляет собой матрицу 2 × 2, заданную:

$$A = \begin{pmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{pmatrix} \quad (2.3)$$

Чтобы смоделировать n шагов общей модели DSMC, можно определить n-ступенчатую матрицу перехода A (n) как:

$$A_{ij}(n) := p(X_{t+n} = j \mid X_t = i) \quad (2.4)$$

Нетрудно показать, что A (m + n) = A (m) A (n) и, следовательно, A (n) = A (1) n. Это означает, что n шагов модели DSMC можно моделировать просто путем повторного умножения матрицы перехода на себя.

2.2.6 Скрытые марковские модели

Скрытые марковские модели - это марковские модели, в которых состояния теперь «скрыты» из вида, а не непосредственно наблюдаемы. Вместо этого существует набор выходных наблюдений, связанных с состояниями, которые непосредственно видны. Чтобы сделать это конкретным для количественного финансового примера, можно думать о государствах как о скрытых «режимах», при которых рынок может действовать, а наблюдения - это доходы от активов, которые непосредственно видны.

В марковской модели необходимо создать совместную функцию плотности для наблюдений. Была указана матрица перехода по времени, которая позволяет полностью моделировать модель. Для скрытых марковских моделей необходимо создать набор дискретных состояний zt∈ {1, ..., K} (хотя для целей обнаружения режима часто требуется только K≤3) и моделировать наблюдения с дополнительной вероятностью модель, p (xt|zt). То есть условная вероятность увидеть конкретное наблюдение (возврат активов) при условии, что состояние (рыночный режим) в настоящее время равно zt.

В зависимости от указанного состояния и вероятности перехода наблюдения, скрытая марковская модель будет стремиться оставаться в определенном состоянии, а затем внезапно перейти в новое состояние и некоторое время оставаться в этом состоянии. Именно такое поведение желательно от такой модели при попытке применить ее к рыночным режимам. Ожидается, что сами режимы не будут слишком быстро меняться (рассмотрите нормативные изменения и другие медленные макроэкономические эффекты). Однако, когда они меняются, они, как ожидается, сохраняются в течение некоторого времени [25].

2.2.7 Математическая спецификация скрытой марковской модели

Соответствующая функция плотности соединения для HMM дается (опять же с использованием обозначения от Murphy (2012))

$$p(z_{1:T} \mid x_{1:T}) = p(z_{1:T})\, p(x_{1:T} \mid z_{1:T}) = \left[p(z_1) \prod_{t=2}^{T} p(z_t \mid z_{t-1}) \right] \left[\prod_{t=1}^{T} p(x_t \mid z_t) \right] \quad (2.5)$$

В первой строке это указывает на то, что совместная вероятность видеть полный набор скрытых состояний и наблюдений равна вероятности простого наблюдения за скрытыми состояниями, умноженными на вероятность наблюдения наблюдений, обусловленных состояниями. Это имеет смысл, поскольку наблюдения не могут влиять на состояния, но скрытые состояния косвенно влияют на наблюдения.

Вторая строка разделяет эти два распределения на функции перехода. Функция перехода для состояний дается выражением p (zt|zt-1), а для наблюдений (которые зависят от состояний) дается формулой:

$$p(x_t \mid z_t) \quad (2.6)$$

Как и в приведенном выше описании марковской модели, для целей этой статьи предполагается, что как функция перехода состояния, так и наблюдения являются временными. Это означает, что можно использовать матрицу перехода состояний K × K как и раньше с марковской моделью для этого компонента модели.

Однако для рассматриваемого здесь приложения, а именно наблюдений за возвратами активов, значения фактически являются непрерывными. Это означает, что выбор модели для функции перехода наблюдения является более сложным. Общим выбором является использование условного многомерного гауссовского распределения со средним μk и ковариацией σk. Это формализовано ниже:

$$p(x_t \mid z_t = k, \theta) = \mathrm{N}(x_t \mid \mu_k, \sigma_k) \quad (2.7)$$

То есть, если состояние zt в настоящее время равно k, то вероятность видеть наблюдение xt, заданное параметрами модели θ, распределяется как многомерный гуассиан.

Чтобы сделать это немного яснее, следующая диаграмма показывает эволюцию состояний zt и то, как они косвенно приводят к эволюции наблюдений, xt [26]:

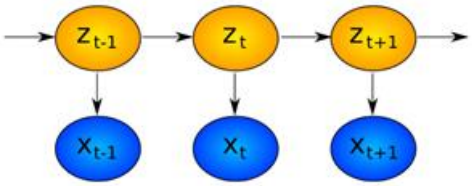

Рисунок 2.12 – Скрытая цепь Маркова: состояния и наблюдения

2.2.8 Фильтрация скрытых марковских моделей

При заданной совместной функции плотности остается рассмотреть, как будет использоваться модель. В общем моделировании состояния пространства часто встречаются три основные задачи: фильтрация, сглаживание и предсказание. В предыдущей статье, касающейся моделей состояния и фильтра Калмана, это кратко описано. Они будут повторяться здесь для полноты:

– Прогнозирование – прогнозирование последующих значений состояния

– Фильтрация – оценка текущих значений состояния из прошлых и текущих наблюдений

– Сглаживание. Оценка прошлых значений состояния с учетом наблюдений.

Фильтрация и сглаживание аналогичны, но не идентичны. Сглаживание связано с желанием понять, что случилось с состояниями в прошлом с учетом текущих знаний, тогда как фильтрация связана с тем, что происходит с государством прямо сейчас.

В рамки этой статьи выходит подробное описание алгоритмов, разработанных для фильтрации, сглаживания и прогнозирования. Основная цель этой серии статей - применить скрытые марковские модели к обнаружению режима. Следовательно, поставленная задача становится определяющей, каков нынешний «рыночный режим» мира в использовании доступных активов. Таким образом, это проблема фильтрации.

Математически условной вероятностью состояния в момент времени t, заданного последовательностью наблюдений до момента времени t, является объект, представляющий интерес. Это включает в себя определение p (zt|x1: T). Как и в случае фильтра Kalman, можно рекурсивно применить правило Байеса, чтобы добиться фильтрации на HMM [26].

2.3 Используемые методы

На основании вышесказанного в работе могут быть использован следующий подход:

1. Непрерывное вейвлет-преобразование (CWT) или преобразование постоянного качества (constant-q) для преобразования звукозаписи в частотно-временное представление вместо STFT.
2. Получение векторов признаков, как признаки учитывать также MFCC.
3. RNN-RBM для обучения признаков.
4. 88 SVM-классификаторов One-vs-All.
5. Пост-процессинг скрытыми марковскими моделями.
6. Взаимодействие методов показано на рисунке:

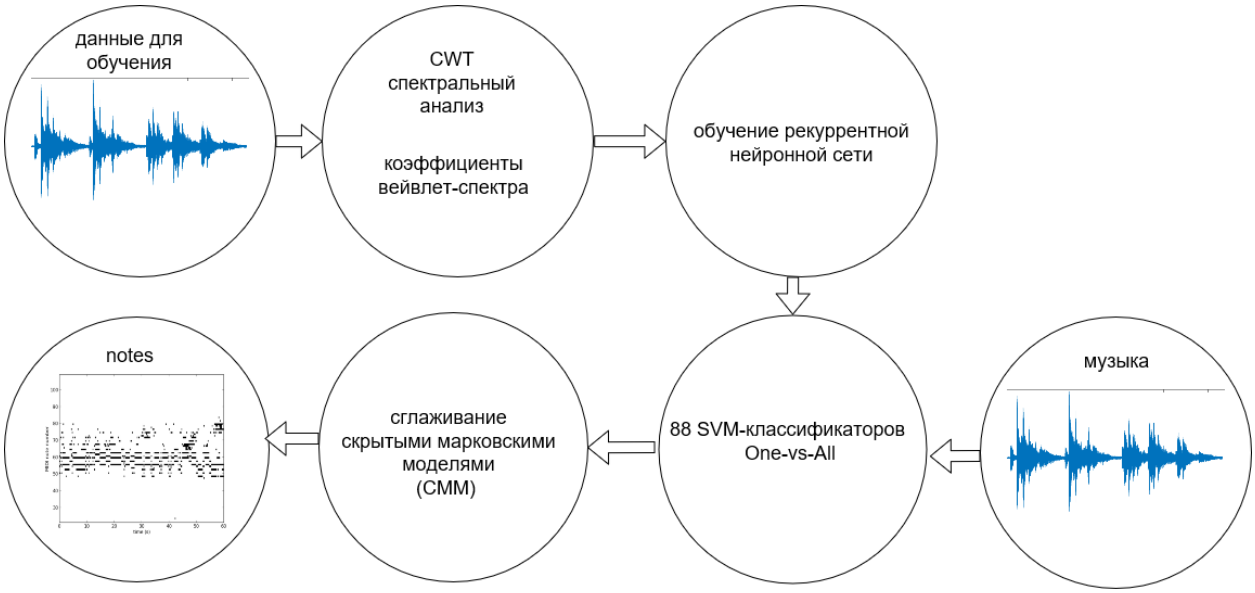

Рисунок 2.13 – Предложенный подход

ГЛАВА 3
ЭКСПЕРИМЕНТАЛЬНЫЙ РАЗДЕЛ

В этом разделе рассмотрим основные шаги, примененные при проведении экспериментов.

Обучение производилось на видеокарте Nvidia Geforce GTX 850m с 2 ГБ оперативной памяти.

Проект выложен в открытый доступ на платформе github [30].

Необходимые требования для запуска приложения:

1. Python 3.5 либо выше. Желательно наличие видеокарты Nvidia.
2. TensorFlow, Theano, или CNTK как бэкенд для Keras.
3. Nvidia software: NVIDIA GPU drivers, CUDA Toolkit, cuDNN SDK.
4. Python пакеты tensorflow-gpu, h5py, keras, pretty_midi, numpy, matplotlib, scipy, yaml, math, librosa, sklearn.

Назначения некоторых python пакетов:

– LibROSA – это пакет для анализа музыки и аудио, он обеспечивает строительные блоки, необходимые для создания музыкальных информационно-поисковых систем;

– Pretty_midi содержит служебные функции и классы для обработки MIDI-данных;

– Numpy – это фундаментальный пакет для научных вычислений на python, он содержит среди прочего: мощный N-мерный массив и сложные функции к нему;

– Matplotlib – библиотека для построения 2D графиков;

– Scipy – основанная на Python экосистема программного обеспечения с открытым исходным кодом для математики, науки и техники;

– Sklearn – библиотека машинного обучения;

Процесс обучения состоит из следующих шагов, см рисунок 3.1:

– загрузка и подготовка данных;

– определение сети;

– компиляция модели;

– приспосабливание модели (fit);

– оценка модели (evaluate);

Структура проекта имеет вид:

– datasets – каталог с выделенными данными для обучения, для каждого произведения имеется по 2 файла со свойствами и метками в формате .npy, готовых для загрузки в numpy массив;

– samples – произведения в форматах mp3 и ogg, а также midi файлы к

этим произведениям;

– trained_models – сохраненные натренированные модели (конфигурация и веса) в форматах json и h5;

– train.py – скрипт, содержащий определение архитектуры модели и шаги по ее обучению;

– test.py – скрипт, содержащий код для вызыва предсказаний на тестовых данных и сравнения предсказаний с метками на графике;

– data_generator.py содержит определение для генератора данных, он нужен, так как данные не помещаются в оперативную память;

– correlate_train_data.py – скрипт, определяющий относительное смещение между частотно-временными свойствами и метками для произведения из обучающей выборки;

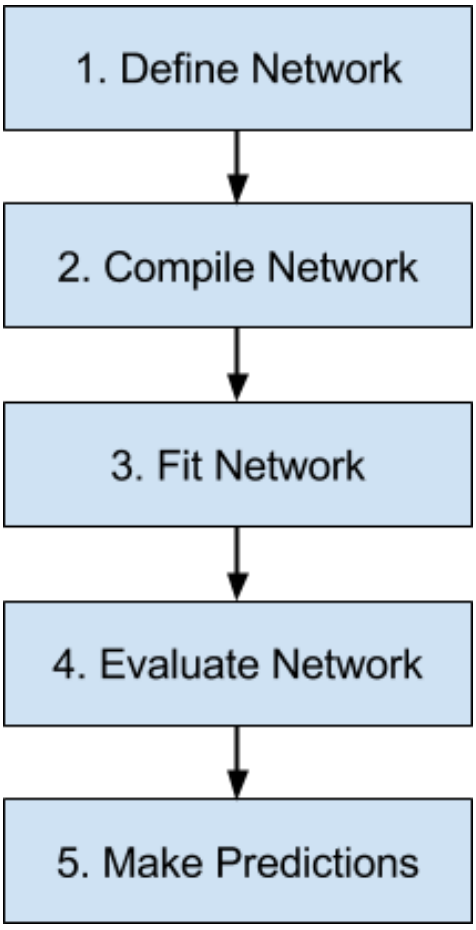

Рисунок 3.1 – Шаги обучения модели

3.1 Подготовка данных

Датасет произведений фортепиано и соответствующие им midi записи

находятся на странице Classical piano dataset [31]. Здесь представлены произведения двадцати пяти известных композиторов. Датасет содержит музыкальные файлы разных форматов классической музыки. Пьесы наигрываются на цифровом пианино с помощью секвенсора на базе MIDI, а затем конвертируются в аудиоформаты.

Для обучения было взято 305 произведений.

Так как используется подход обучение с учителем, то на вход сети подаются, как и временно частотные фреймы данных, так и метки – правильные ответы, к которым сеть будет стремиться во время обучения, они представлены в формате midi.

Аудио приводится в моно формат и над ним производится Constant-Q преобразование. Далее, произведения разделяются на куски, размером в 512 кадров каждый. Данные для распознавания тоже должны быть такой размерности. Для Constant-Q преобразования используется библиотека librosa - https://librosa.github.io/librosa/. Она содержит реализацию алгоритмов для аудио-анализа. Преобразование выполняется с помощью метода

```
librosa.core.cqt(y, sr=22050, hop_length=512, fmin=None, n_bins=84, bins_per_octave=12, tuning=0.0, filter_scale=1, norm=1, sparsity=0.01, window='hann', scale=True, pad_mode='reflect')
```

где

– `y` – временной ряд аудио с сигнатурой `np.ndarray [shape=(n,)]`, `n` – количество временных отсчетов в произведении;

– `sr` – частота дискретизации `y`;

– `n_bins` – количество частотных интервалов, начиная с `fmin`.

– `bins_per_octave` - кол-во частотных интервалов для каждой октавы (всего 7 октав и одна неполная);

– возвращаемое значение – преобразование Constant-Q для каждой частоты в каждый временной отсчет, сигнатура: `np.ndarray [shape=(n_bins, t), dtype=np.complex or np.float]`

Частота дискретизации выбрана стандартной для CD – 44100 Гц. Некоторые методы музыкальной транскрипции снижают частоту дискретизации до 22050 Гц для уменьшения кол-ва данных, но это также снижает точность преобразования, так как снижают количество, а, следовательно, и качество данных.

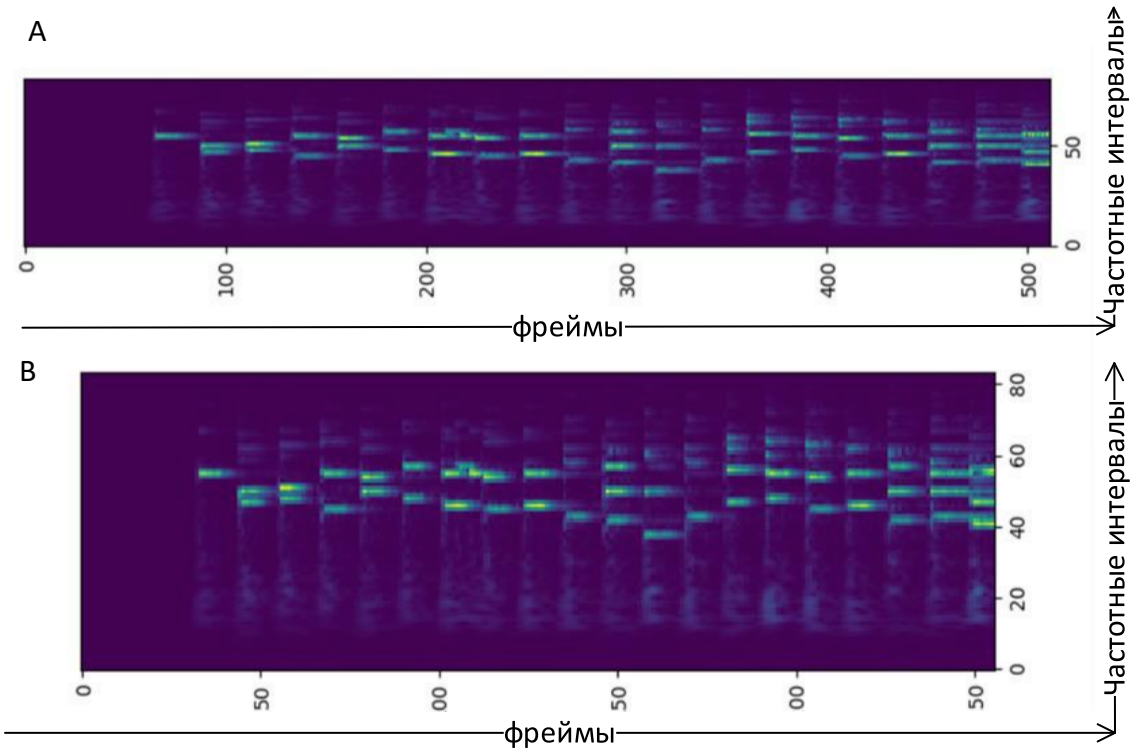

Рисунок 3.2 – Сравнение временно-частотного преобразования с различной частотой дискретизации

На рисунке 3.2B частота дискретизации – 22050 Гц, на графике 3.2A – 44100 Гц.

Кол-во частотных интервалов берется как значение по умолчанию - 84. Количество частотных интервалов на октаву - 12, тоже по умолчанию. Таким образом единица данных для обучения имеет размерность 512*84 (фреймы*частотные интервалы)

Возможно, что увеличение разрешения по частотным интервалам благоприятно скажется на результатах классификации. Рассчитаем увеличенное кол-во частотных интервалов, при условии, что все частоты фортепиано должны присутствовать после преобразования:

$$n\_bins = bins\_per\_octave * \text{n\_octaves} = 16 * 7 = 112, \qquad (3.1)$$

Допустим, что для улучшения частотного разрешения достаточно 16 вместо 12 частотных интервалов на октаву см. формулу 3.1.

На рисунке 3.3A преобразование с стандартными 84 частотными интервалами и 12 интервалами на октаву, 33B с 112 частотными интервалами и 16 интервалами на октаву. При более детальном рассмотрении можно увидеть, что представление информации улучшилось, но при этом увеличилась размерность данных для обучения.

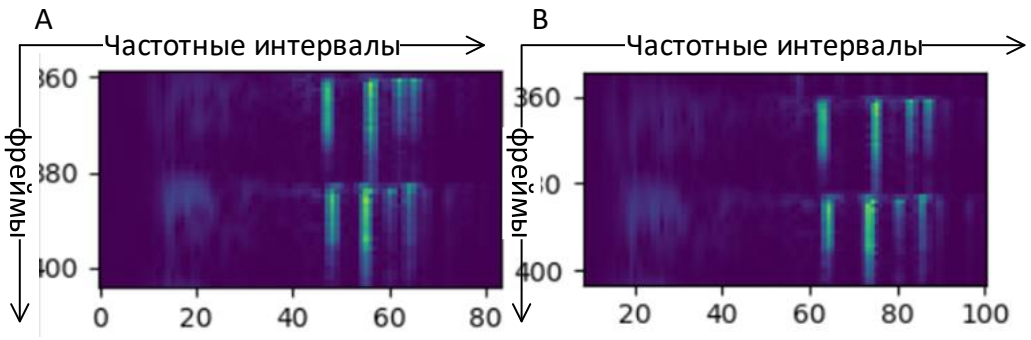

Рисунок 3.3 – Сравнение временно-частотного преобразования с различным кол-вом частотных интервалов на октаву

Метод по получению свойств для обучения из аудио файла выглядит так:

```
def cqt_features(audio_filename):
    y, sr = librosa.load(audio_filename, 44100)
    y_mono = librosa.to_mono(y)
    spectrum = np.abs(librosa.cqt(y_mono, sr=44100))
    spectrum = np.transpose(spectrum)
    return normalize(spectrum)
```

В коде делаются следующие преобразования:

5. Загрузка аудиофайла в виде временного ряда, значения с плавающей запятой. Аудио дискретизировано с частотой 44100 Гц.
6. Аудио преобразуется в моно-формат.
7. Constant-Q преобразование. В качестве свойств для обучения используются значения амплитуд временно-частотного преобразования.
8. Транспонирование – приведение данных к матрице формата 512x84.
9. Нормализация данных, приведение к диапазону от 0 до 1.

Для представления меток (правильных классов), соответствующих частотно-временному представлению используется multi-hot кодирование так как одному фрейму сопоставляется 88 отдельных классов нот, индицирующих наличие или отсутствие каждой из нот в данном фрейме, см рисунок 3.4.

frames \ notes	0	1	2	…	86	87
0	0	0	1	…	1	0
1	0	0	1	…	1	1
2			labels			
…	1	1	1	…	1	0
511	1	0	1	…	0	0

Рисунок 3.4 – Multi-hot кодирование

На рисунке 3.5 приведен пример кусочка тренировочных данных, размером 512 фреймов. На 3.5B показано временно-частотное представление, на нем светлыми полосами видны частоты, задействованные при проигрывании записи. На 3.5A видны номера нот (от 0 до 87) и их продолжительность.

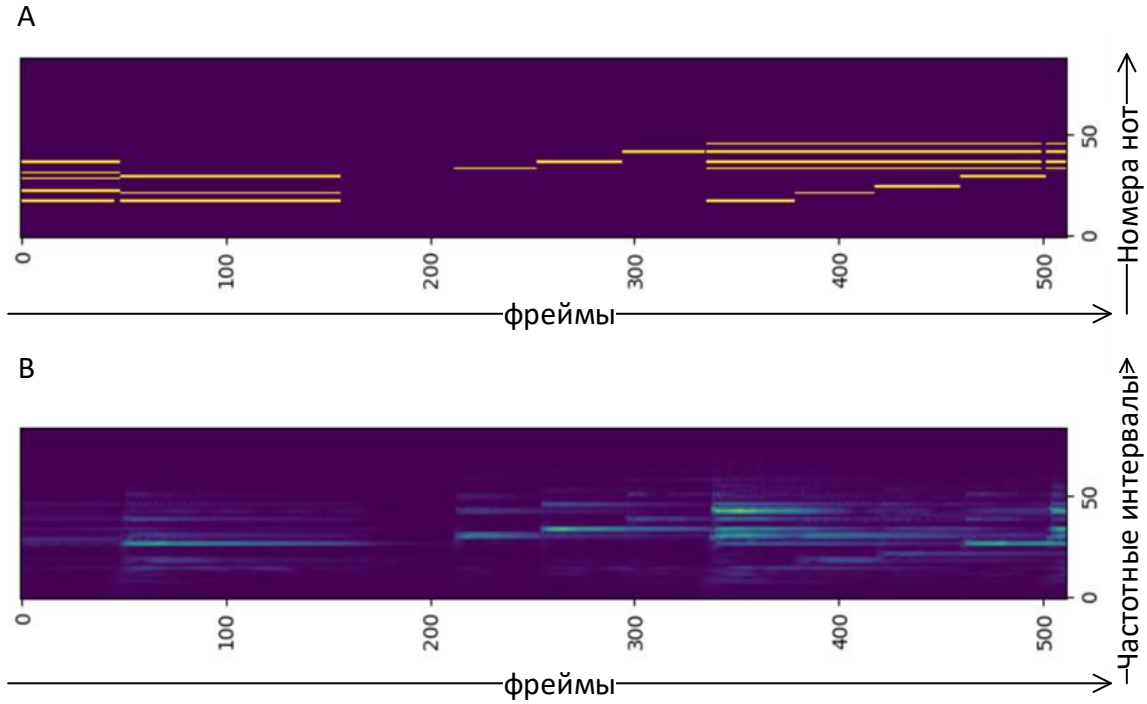

Рисунок 3.5 – Тренировочные данные

Для тренировочных данных сделаны следующие допущения:

1. Громкость проигрываемых нот не учитывается при обучении.
2. Не учитываются физические характеристики проигрываемой ноты, такие как затухание.

Модель Attack/Delay/Sustain/Release (ADSR) показывает, как музыкальное событие, такое как как щипок струны, меняется со временем. Модель продемонстрирована на рисунке 3.6. Когда акустический музыкальный инструмент производит звук, громкость и спектральный состав звука меняются со временем разными способами, которые варьируются от инструмента к инструменту. «Атака» и «затухание» звука иметь большое влияние на звуковой характер инструмента. Сеть должна будет научиться распознавать разные этапы звука, чтобы определить, когда определенная нота нажата [29].

Система фокусирует внимание на событиях начала и окончания ноты, что позволяет упросить модель.

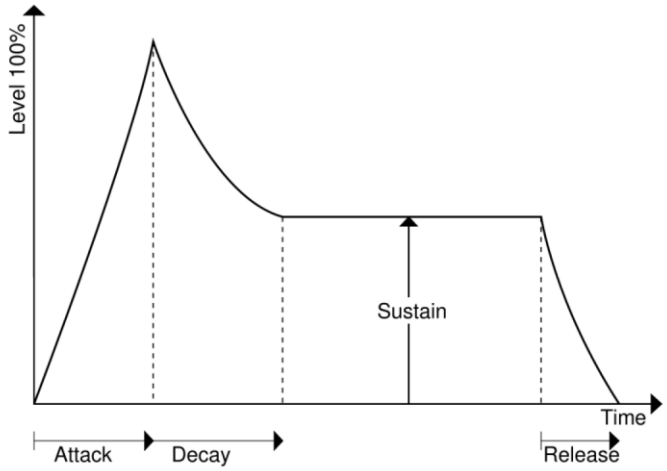

Рисунок 3.6 – Модель Attack/Delay/Sustain/Release (ADSR)

Данные в датасете не выровнены по времени, поэтому выравнивание нужно производить до обучения, чтобы частотные фреймы точно соответствовали нотам, полученным из midi. Пример не выровненных данных на рисунке 3.7.

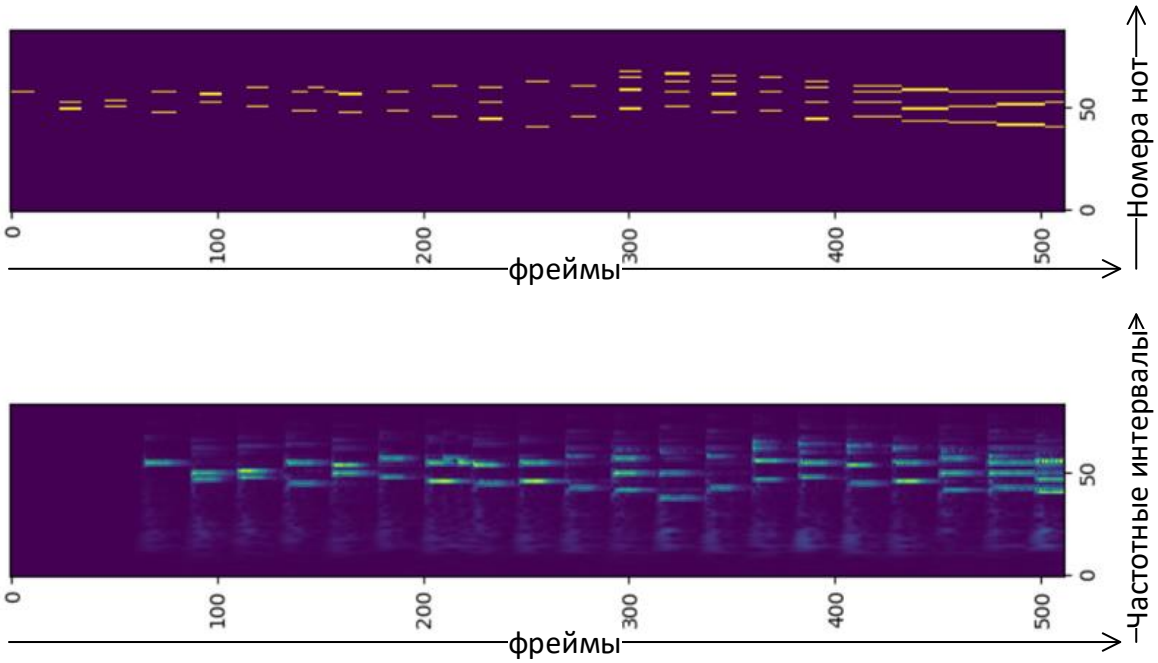

Рисунок 3.7 – Пример данных, не синхронизированных по времени

Для выравнивания данных используется корреляция. Корреляция считается последовательно для всех кусочков произведения. Так как данные представляют из себя не одномерный сигнал, для подсчета корреляции их нужно привести в одномерный вид. Для этого все значения сигнала суммируются для каждого отсчёта времени, и корреляция считается по одномерной сумме сигналов. Рисунок 3.8В.

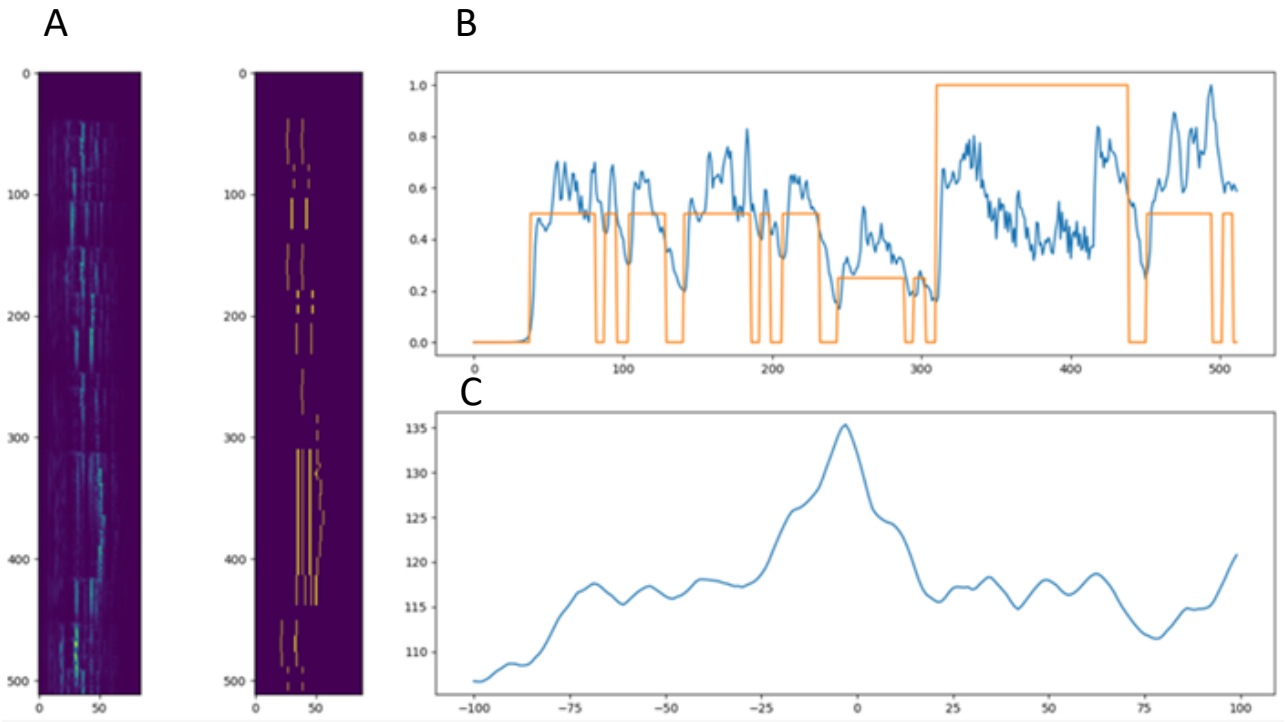

Рисунок 3.8 – Пример хорошо коррелированных данных

На рисунке 3.8A видны тренировочные данные, 3.8B – их суммированные плоские наложения, 3.8C их корреляция. Из рисунка видно, что относительное смещение данных равно -3 фрейма.

Не во всех случаях корреляционный анализ позволяет понять настоящее смещение данных. Рисунки 3.9 и 3.10 демонстрируют плохо коррелированные данные. Для обучения используются только хорошо коррелированные фрагменты - это, примерно, 5000 фрагментов по 512 фрейма.

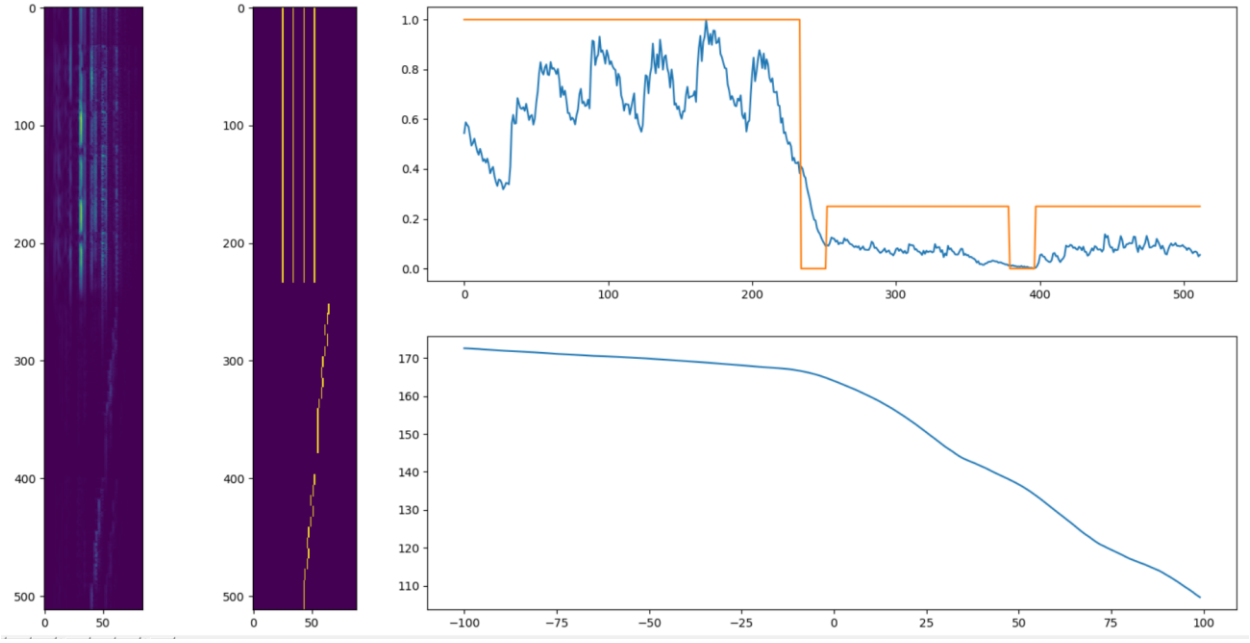

Рисунок 3.9 – Пример плохо коррелированных данных

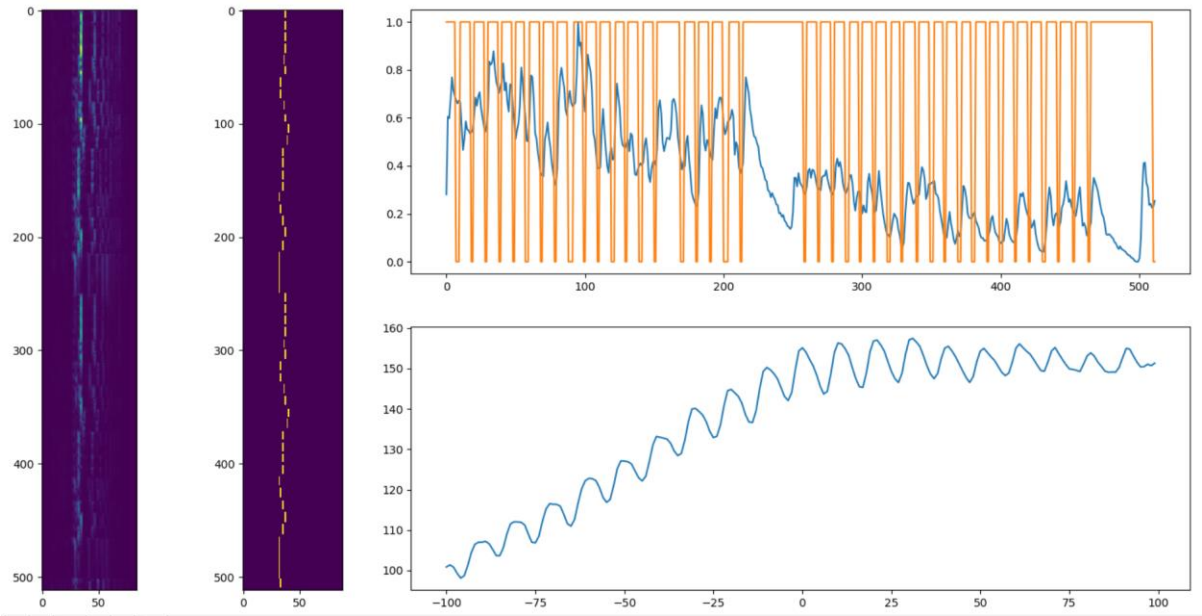

Рисунок 3.10 – Пример плохо коррелированных данных

Обучающая выборка слишком большая, чтобы поместиться в оперативную память, поэтому используется генератор, который последовательно подает на вход сети данные, читая их с диска. Чтобы реализовать генератор в keras нужно наследоваться от класса keras.utils.Sequence. Каждая генератор должнен реализовывать методы __getitem__ и __len__. Если набор данных необходимо модифицировать между эпохами обучения, можно реализовать метод on_epoch_end. Метод __getitem__ должен возвращать полный батч. Keras.utils.Sequence - более безопасный способ сделать многопроцессорную обработку. Эта структура гарантирует, что сеть будет обучаться только один раз для каждой выборки за эпоху.

Таким образом правильная подготовка данных к обучению – это ключевой фактор в обучении. Неправильная подготовка данным может сделать последующее обучение менее результативным, или совсем невозможным. Чтобы улучшить результаты обучения нужно в первую очередь подумать, как улучшить качество и или представление данных. Например, не выровненные по времени частотно-временные характеристики к меткам, делают дальнейшее обучение невозможным.

3.2 Гиперпараметры

Гиперпараметры — это значения, которые нужно подбирать вручную и зачастую методом проб и ошибок. Среди таких значений можно выделить [28]:

– момент и скорость обучения;

– количество скрытых слоев;
– количество нейронов в каждом слое;
– процент временно исключенных нейронов из рекуррентного слоя, для избегания переобучения;

Также опытным путем необходимо подобрать количество эпох обучения.

Произошла одна эпоха (epoch) — весь датасет прошел через нейронную сеть в прямом и обратном направлении только один раз. Нужно помнить, что мы используем ограниченный датасет, чтобы оптимизировать обучение и подстроить кривую под данные. Делается это с помощью градиентного спуска — итеративного процесса. Поэтому обновления весов после одного прохождения недостаточно. Одна эпоха приводит к недообучению, а избыток эпох — к переобучению.

3.3 Модели

Для глубокого обучения использовался каркас – Keras. Базовая структура данных Keras - это модель, способ организации слоев. Самым простым типом модели является последовательная модель (Sequential) - линейный стек слоев. Добавить слой в модель можно при помощи метода add.

Модель была получена эмпирическим путем, посредством подбора ее архитектуры и гиперпараметров.

Метрикой успешности сети является ошибка (Loss). Ошибка — это процентная величина, отражающая расхождение между ожидаемым и полученным ответами. Ошибка формируется каждую эпоху и должна идти на спад.

Чем меньшей ошибки удавалось достичь при обучении сети за определенное кол-во итераций, тем успешнее считались подобраны гиперпараметры и, соответственно, более точна модель.

Опишем несколько моделей, которые были обучены. Каждая модель будет пронумерована, чтобы было удобнее обращаться к ней в дальнейшем.

Код модели №1:

```
model = Sequential()
model.add(LSTM(200,
               dropout=0.5,
               recurrent_dropout=0.5,
               input_shape=(512, 84),
```

```
                    return_sequences=True))
    model.add(TimeDistributed(Dense(100, activation='relu')))
    model.add(TimeDistributed(Dense(88, activation='sigmoid')))
    model.compile(loss='binary_crossentropy',optimizer='adam')
```

Первый слой сети – рекуррентный LSTM слой с 200 нейронами. Для предотвращения переобучения LSTM имеет dropout. Dropout – это техника исключения нейронов из сети. Сети для обучения получаются с помощью исключения из сети (dropping out) нейронов с вероятностью p, таким образом, вероятность того, что нейрон останется в сети, составляет q=1-p. “Исключение” нейрона означает, что при любых входных данных или параметрах он возвращает 0. Исключенные нейроны не вносят свой вклад в процесс обучения ни на одном из этапов алгоритма обратного распространения ошибки (backpropagation); поэтому исключение хотя бы одного из нейронов равносильно обучению новой нейронной сети. p = 0.5, это значит, что половина нейронов будет проигнорирована. Dropout применяется как на входе слоя dropout=0.5, так и в циклах самого рекуррентного слоя recurrent_dropout=0.5. LSTM сохраняет временную составляющую входных данных за счет установки флага return_sequences=True.

Второй слой – это полносвязный слой с 100 нейронами и функцией активации relu, используется для интерпретации свойств, извлеченных скрытым слоем LSTM. Слой использует TimeDestributed обертку. Эта обертка применяет слой к каждому временному фрагменту входа.

Последний полносвязный слой с кол-ом нейронов равным кол-ву нот фортепиано (88) служит для классификации. Sigmoid функция активации нужна для бинарной классификации каждой из 88 нот.

Модель использует функцию потерь binary_crossentropy для бинарной классификации каждой из нот.

Оптимизатор – adam. Оптимизаторы - это алгоритмы оптимизации, которые могут использоваться вместо классической процедуры стохастического градиентного спуска для итеративного обновления весов сети на основе обучающих данных. В экспериментах использовались также RMSprop оптимизатор, этот оптимизатор обычно является хорошим выбором для рекуррентных нейронных сетей.

Модель №2 имеет вид:

```
model = Sequential()
model.add(LSTM(146,
               dropout=0.2,
               recurrent_dropout=0.2,
               input_shape=(512, 84),
```

```
                return_sequences=True))
model.add(TimeDistributed(Dense(146, activation='relu')))
model.add(TimeDistributed(Dense(88, activation='sigmoid')))
model.compile(loss='binary_crossentropy',optimizer='adam')
```

По сравнению с моделью №1, модель №2 имеет 146 нейронов в слое LSTM и 146 нейронов в полносвязном слое, также в ней уменьшен процент случайно неиспользуемых связей (dropout) с 0.5 до 0.2.

Модель № 3 имеет вид:

```
model = Sequential()
model.add(LSTM(120,
               dropout=0.2,
               recurrent_dropout=0.2,
               input_shape=(512, 84),
               return_sequences=True))
model.add(LSTM(120,
               dropout=0.2,
               recurrent_dropout=0.2,
               return_sequences=True))
model.add(TimeDistributed(Dense(120, activation='relu')))
model.add(TimeDistributed(Dense(88, activation='sigmoid')))
model.compile(loss='binary_crossentropy',optimizer='adam')
```

В данной модели добавлен еще один рекуррентный слой, а кол-во нейронов в каждом слою снижено до 120.

Модель № 4:

```
model = Sequential()
model.add(LSTM(120,
               dropout=0.2,
               recurrent_dropout=0.2,
               input_shape=(512, 84),
               return_sequences=True))
model.add(TimeDistributed(Dense(120, activation='relu')))
model.add(TimeDistributed(Dense(88, activation='sigmoid')))
model.compile(loss='binary_crossentropy',optimizer='adam')
```

Данная модель имеет один слой с 120 нейронами. Примерное кол-во его нейронов было посчитано по формулам ниже.

Для примерного подсчета возможного кол-ва нейронов скрытого слоя

можно использовать формулу 3.2, однако, кол-во подсчитанных нейронов лучше всего подбирать эмпирическим путем и под конкретную задачу.

$$N_h = \frac{2}{3} \cdot (N_i + N_o) = \frac{2}{3} \cdot (84 + 88) \approx 115 \quad (3.2)$$

где

- N_i - кол-во нейронов на входном слое;
- N_o - кол-во нейронов на выходном слое;
- N_h - кол-во нейронов скрытого слоя;

Другая формула также учитывает размер обучаемых данных:

$$N_h = \frac{N_s}{(\alpha \cdot (N_i + N_o))} = \frac{512 \cdot 84}{(2 \cdot (84 + 88))} \approx 125 \quad (3.3)$$

3.4 Обучение

Для обучения выборка разбита на 3 части:

- тренировочную;
- валидационную;
- тестовую;

Размер тестовой и валидационной выборок значительно меньше, чем тренировочной.

Процесс обучения – значение функции потерь выводится в консоль и пишется в логи tensorboard. Чтобы увидеть график уменьшения функции потерь с ходом обучения, нужно запустить tensorboard:

```
tensorboard –logdir=<log_dir>
```

Сервер tensorboard поднимается на 6006 порту.

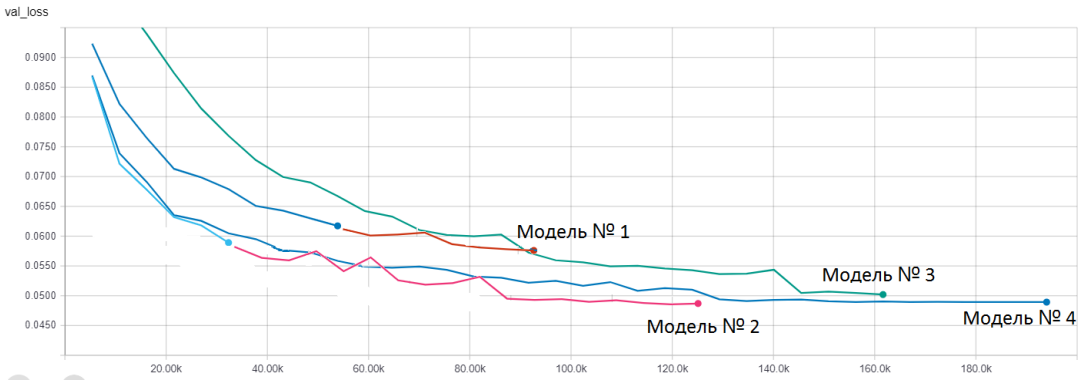

Рисунок 3.11 – Tensorboard визуализация функций потерь для валидационного набора данных при обучении

На рисунке 3.11 видно, как уменьшается функция потерь для валидационной выборки с каждой эпохой обучения, это дает понять, что архитектура сети выбрана правильно, т.е. сеть способна обучаться на имеющихся данных.

Tensorboard позволяет сравнивать архитектуры сетей во время обучения. На рисунке 3.12 показаны графики функций потерь на валидационном наборе при обучении модели № 1 и №2. Из графика видно, что модель №2 даст лучшие результаты после обучения. Функция потерь модели №1 достигает минимального значения равного 0.0627, в то время как для модели №2 – 0.0533 в конце обучения. Таким образом tensorboard позволяет достаточно рано понять, насколько хороша архитектура, не ожидая конца обучения.

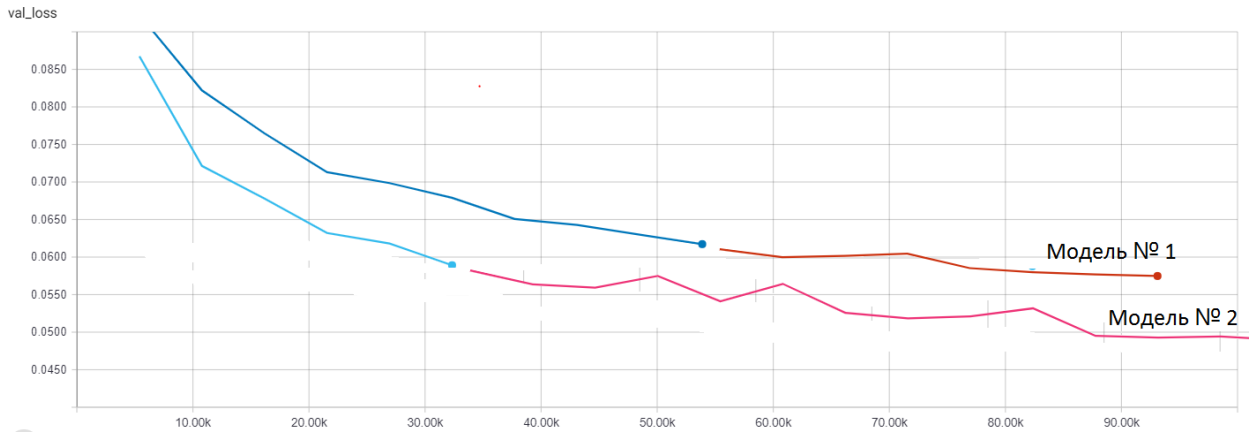

Рисунок 3.12 – Tensorboard график функции потерь для моделей №1 и № 2

На рисунке 3.13 видно, что модель с двумя рекуррентными слоями обучалась дольше. Предполагалась, что ее обобщающая способность будет выше, чем у модели № 4 за счёт более глубоких связей.

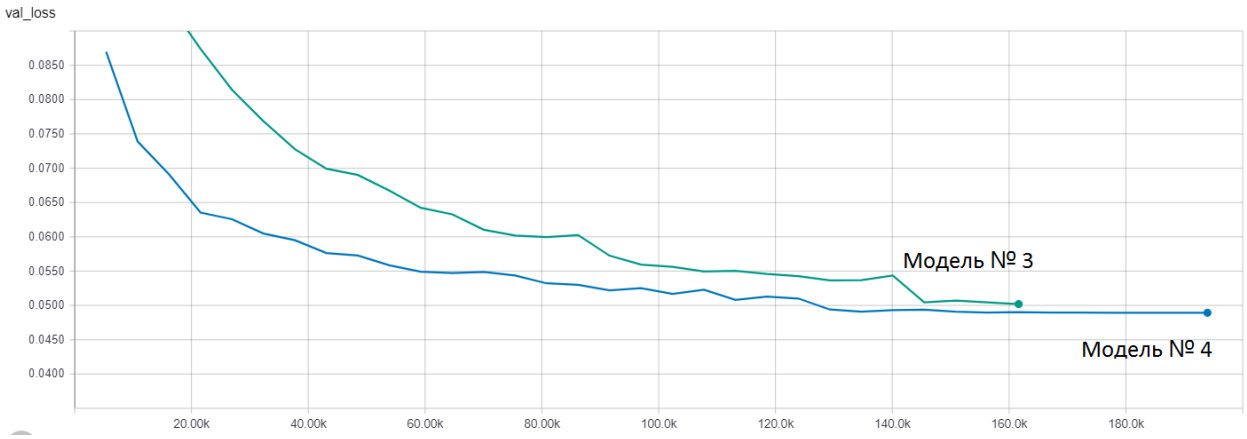

Рисунок 3.13 – Tensorboard график функции потерь для моделей №3 и № 4

Для обучения также используются keras lifecycle callbacks, чтобы иметь возможность влиять на ход обучения после его начала.

ReduceLROnPlateau – позволяет уменьшить скорость обучения, если функция потерь не уменьшается N эпох. Модели часто выигрывают от снижения скорости обучения в 2-10 раз после застоя обучения.

TerminateOnNaN – функция обратного вызова, завершающая обучение при обнаружении значения NaN для функции потерь.

TensorBoard - этот обратный вызов записывает журнал для TensorBoard, который позволяет визуализировать динамические графики показателей обучения и тестирования, а также гистограммы активации для различных слоев в вашей модели.

Функция обратного вызова для сохранения весов в конце каждой эпохи обучения. Это позволяет упростить процесс выбора кол-ва эпох, так как для каждой эпохи будет сохранена версия сети, обученная до этой эпохи. После обучения стоит просто выбрать ту версию сети (весов), у которой функция потерь наименьшая.

Таким образом, такие инструменты как tensorboard и функции обратного вызова упрощают процесс обучения, например, можно решить вопрос о кол-ве эпох следующим образом: выбрать большое кол-во эпох, например, равное 100 и повесить на функцию обратного вызова on_epoch_end сохранение весов модели. Когда по графику tensorboard наблюдается плато в обучении, просто прервать его. После этого остается выбрать из сохраненных весов наиболее подходящие, например, это можно сделать, сохраняя в название файла с весами значение функции потерь, достигнутой на эпохе, после которой эти весы были сохранены. TerminateOnNaN остановит обучение, если функция потерь будет иметь занчение NaN, а ReduceLROnPlateau, например, после достижения какого-то локального минимума, снизит скорость обучения, что позволит выйти из него.

3.5 Результаты обучения

Кривая AUC - ROC - это измерение производительности для задачи классификации при различных настройках порогов. ROC - это кривая вероятности, а AUC - степень или мера отделимости. Качество оценивается как площадь под этой кривой – AUC (AUC = area under the curve). Она говорит, насколько модель способна различать классы. Чем выше AUC, тем лучше модель при прогнозировании 0 как 0 и 1 как 1.

Кривая ROC строится с TPR относительно FPR, где TPR находится на оси y, а FPR на оси x.

Определение терминов, используемых в кривой AUC и ROC.

В случае бинарной классификации есть два класса для каждой из нот: положительный (positive) – нота проигрывается, и отрицательный (negative) – нота не проигрывается.

Тогда на выходе классификатора может наблюдаться четыре различных ситуации:

– Если результат классификации положительный, и истинное значение тоже положительное, то речь идет об истинно-положительном значении (true-positive, TP)

– Если результат классификации положительный, но истинное значение отрицательное, то речь идет о ложно-положительном значении (false-positive, FP)

– Если результат классификации отрицательный, и истинное значение тоже отрицательное, то речь идет об истинно-отрицательном значении (true-negative, TN)

– Если результат классификации отрицательный, но истинное значение положительно, то речь идет о ложно-отрицательном значении (false-negative, FN)

$$TPR\ \left(True\ Positive\ Rate\right)\ /\ Отзыв\ /\ Чувствительность = \frac{TP}{TP + FN}, \qquad (3.1)$$

$$Specificity = \frac{TN}{TN + FP}, \qquad (3.2)$$

$$FPR = 1 - Specificity = \frac{FP}{TN + FP}, \qquad (3.4)$$

Отличная модель имеет AUC близко к 1, что означает, что она имеет хорошую меру отделимости. У плохой модели AUC близко к 0, что означает, что она имеет худшую меру отделимости. Фактически это означает, что она инвертирует результат. Она предсказывает 0-вые значение как 1 и 1-ные как 0. И когда AUC составляет 0.5, это означает, что модель не имеет никакой способности разделения классов.

Давайте интерпретировать вышеизложенные утверждения.

Как известно, ROC является кривой вероятности. Итак, давайте построим распределения этих вероятностей:

Примечание. Красная кривая распределения относится к положительному классу (нота проиграна), а зеленая кривая распределения относится к отрицательному классу (нона не проиграна).

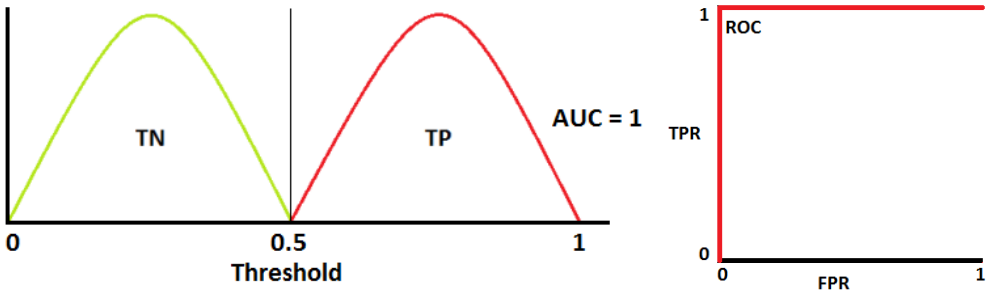

Рисунок 3.14 – Идеальное разделение и ROC кривая

Это идеальная ситуация. Когда две кривые вообще не перекрываются, значит, модель имеет идеальную меру отделимости. Она прекрасно умеет различать положительный класс и отрицательный класс.

Когда два распределения перекрываются, мы вводим ошибки типа 1 и 2. В зависимости от порога мы можем минимизировать или максимизировать их. Когда AUC составляет 0,7, это означает, что есть вероятность того, что модель сможет отличить положительный класс от отрицательного.

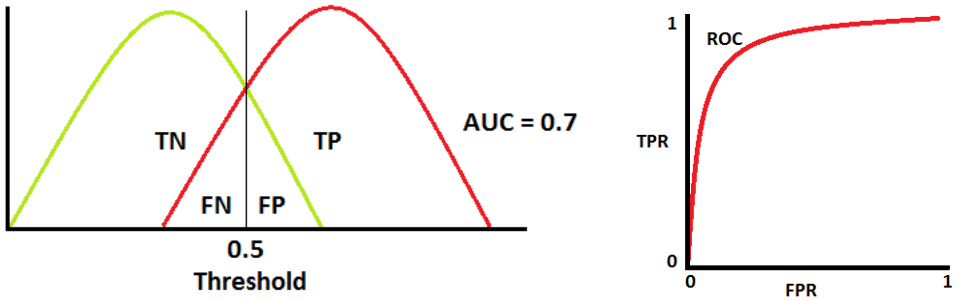

Рисунок 3.15 – Распределения перекрываются

Худшая ситуация, когда AUC составляет приблизительно 0,5, модель не обладает способностью различать положительный класс и отрицательный класс.

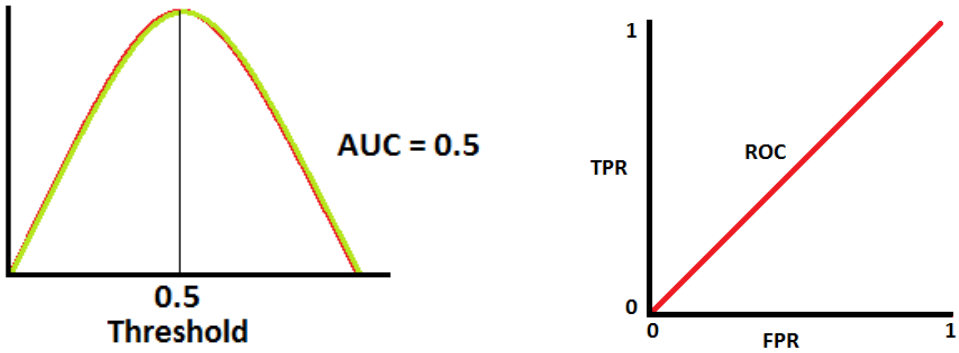

Рисунок 3.16 – Модель не различает классы

Когда AUC составляет приблизительно 0, модель фактически инвертирует предсказания. Это означает, что модель предсказывает

отрицательный класс как положительный класс и наоборот.

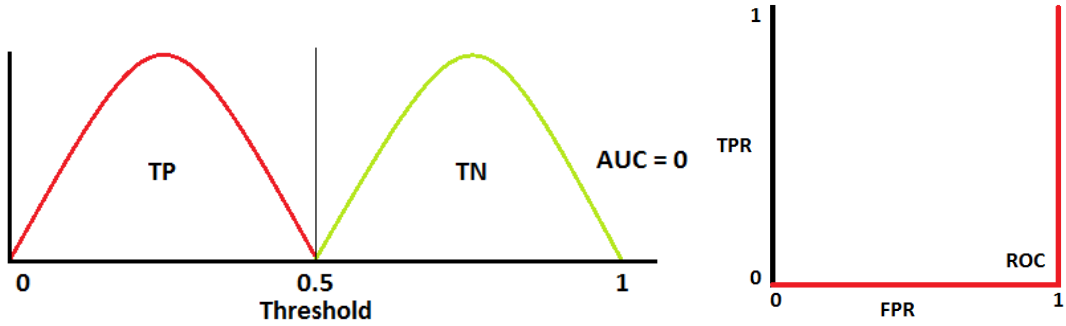

Рисунок 3.17 – Модель инвертирует классы

Построим ROC кривые для некоторых нот из тестового набора, изображенного на рисунке 3.18 и посчитаем AUC для них [27]:

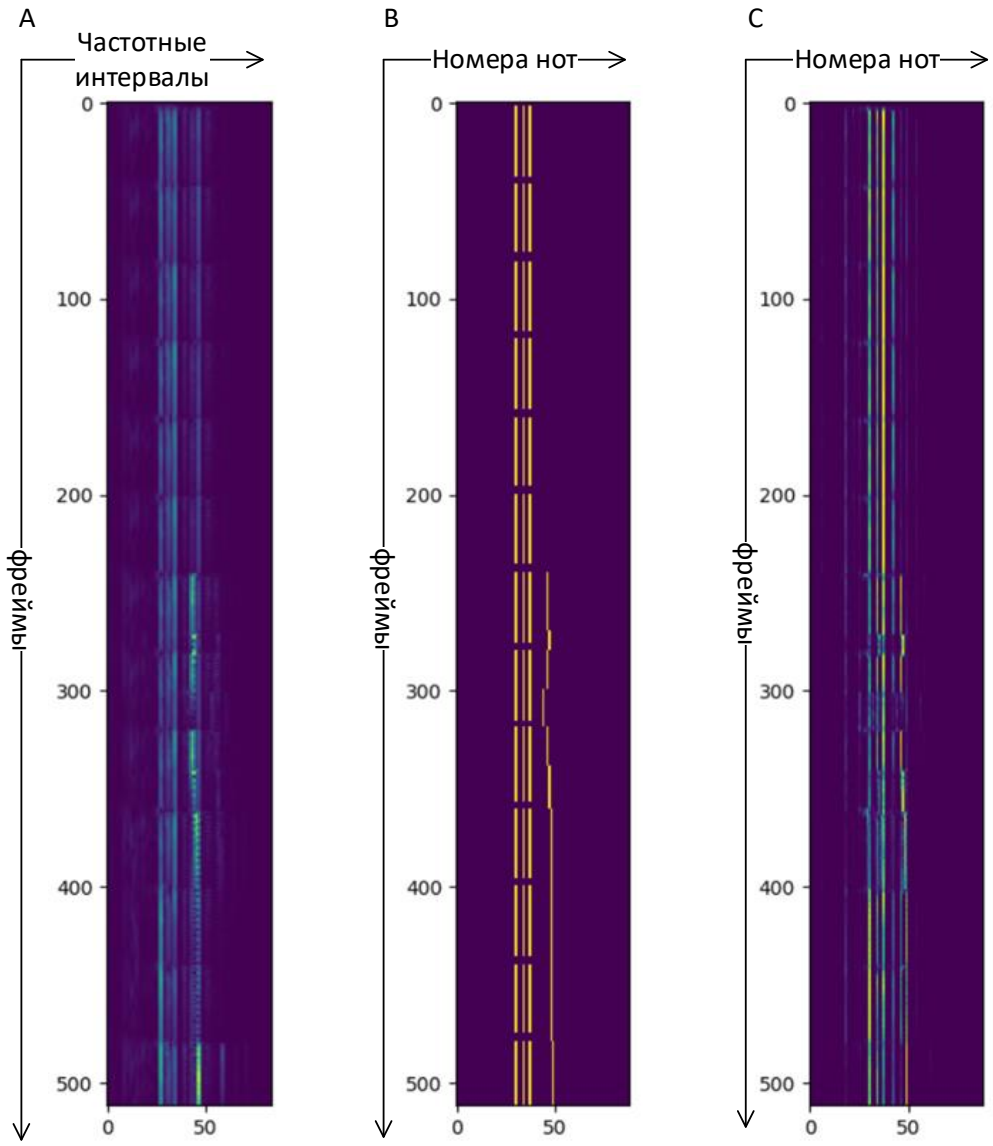

Рисунок 3.18 – Результаты классификации тестового примера №1

На рисунке 3.18A – временно-частотная характеристика участка записи размером 512 фреймов, 3.18B – midi лейблы, позволяющие увидеть правильные классы (правильное присутствие нот), 3.18C – результат классификации, распознанные ноты.

Таблица 3.1 содержит значения AUC для некоторых нот из данной тестовой выборки. Из таблицы видно, что распознающая способность отличается для разных нот. Это объясняется тем, что каждая обучается отдельно, при компиляции модели используется binary_crossentropy функция потерь. Это позволяет обучить только одну нейронную сеть, для решения задачи классификации всех 88 нот вместе взятых.

Таблица 3.1 – AUC для некоторых нот

Номер ноты	AUC
30	0.535
34	0.838
37	0.690
46	0.957
47	0.955
48	0.910
49	0.996

На рисунке 3.14, демонстрирующем ROC кривые для отдельных нот также видно, что распознающая способность нот сильно отличается, это может быть связанно с разным кол-вом нот, поданных на обучения для разных клавиш. ROC кривые можно построить только для тех нот, которые имеют два класса на тестовой выборке, это значит, что нота должна быть проиграна, чтобы для нее построить ROC-кривую. Ниже, на рисунке 3.13 показано распределение кол-ва нот по их номерам в датасете.

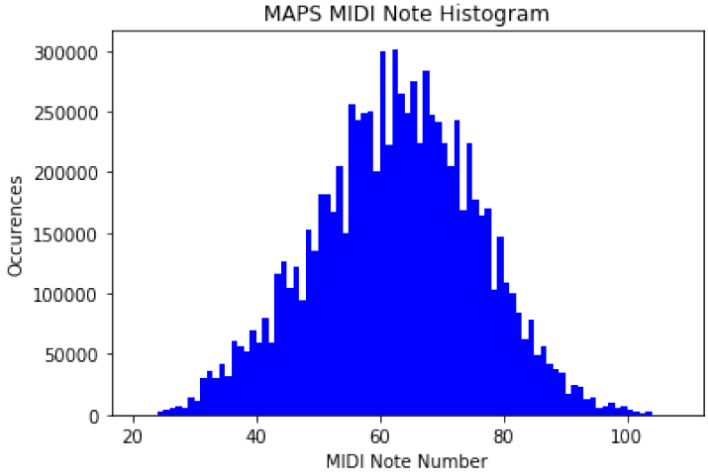

Рисунок 3.13 – Частотное распределение нот в выборке

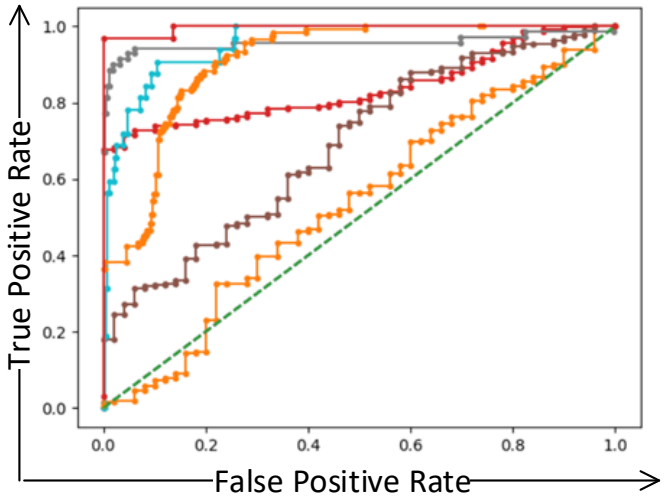

Рисунок 3.14 – Пример ROC кривых для бинарной классификации нот с номерами 30, 34, 37, 46, 47, 48, 49

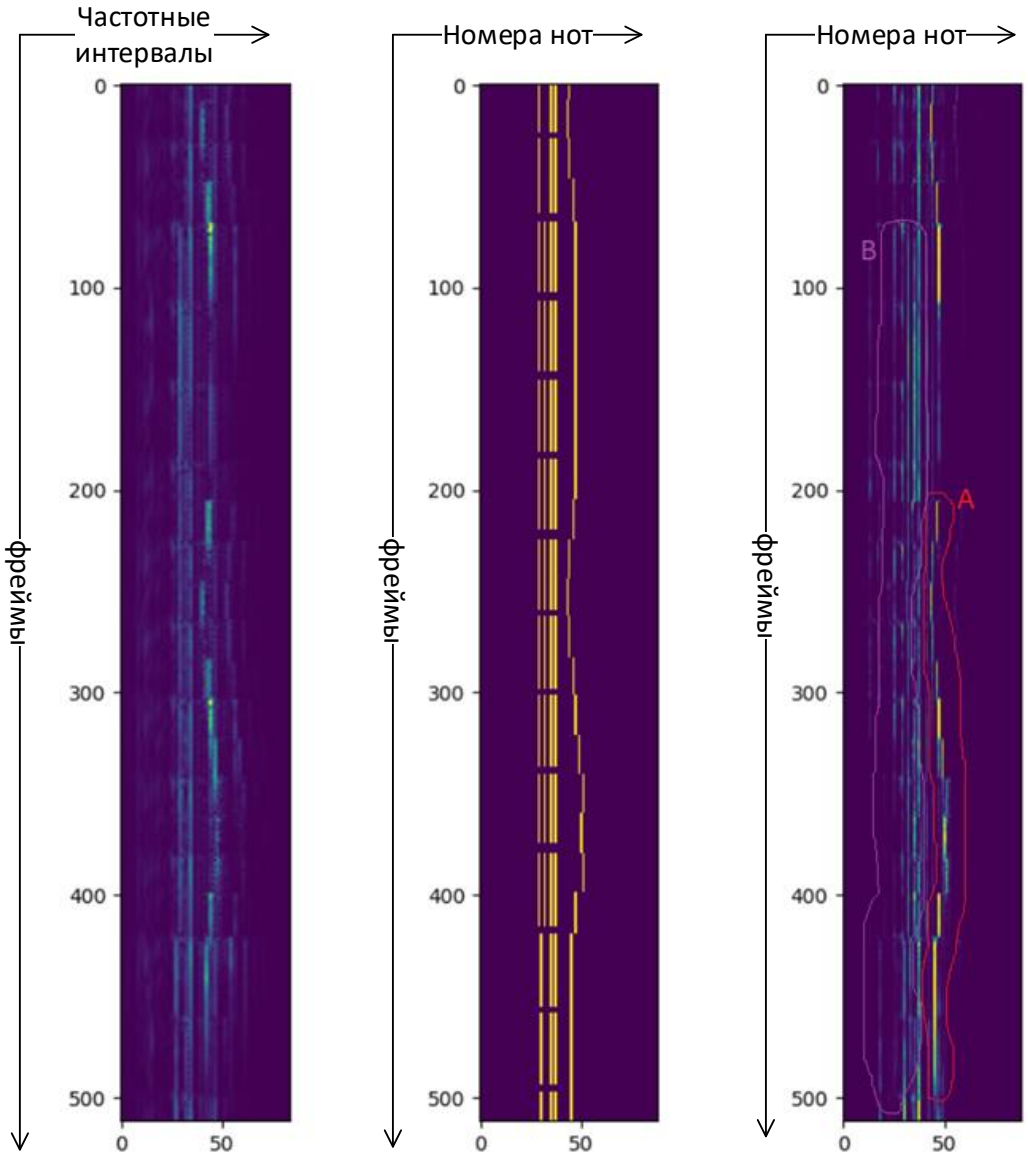

Рисунок 3.15 – Результаты классификации тестового примера №2

На рисунке 3.15 показан результат классификации тестового примера № 2. Из него видно, что область A распознана достаточно хорошо, в этой области находятся более высокие частоты и она, обычно играется правой рукой. Область B практически полностью не распознана. В области B на графике с метками видны повторяющиеся аккорды. Они занимают более низкие частоты и играются левой рукой.

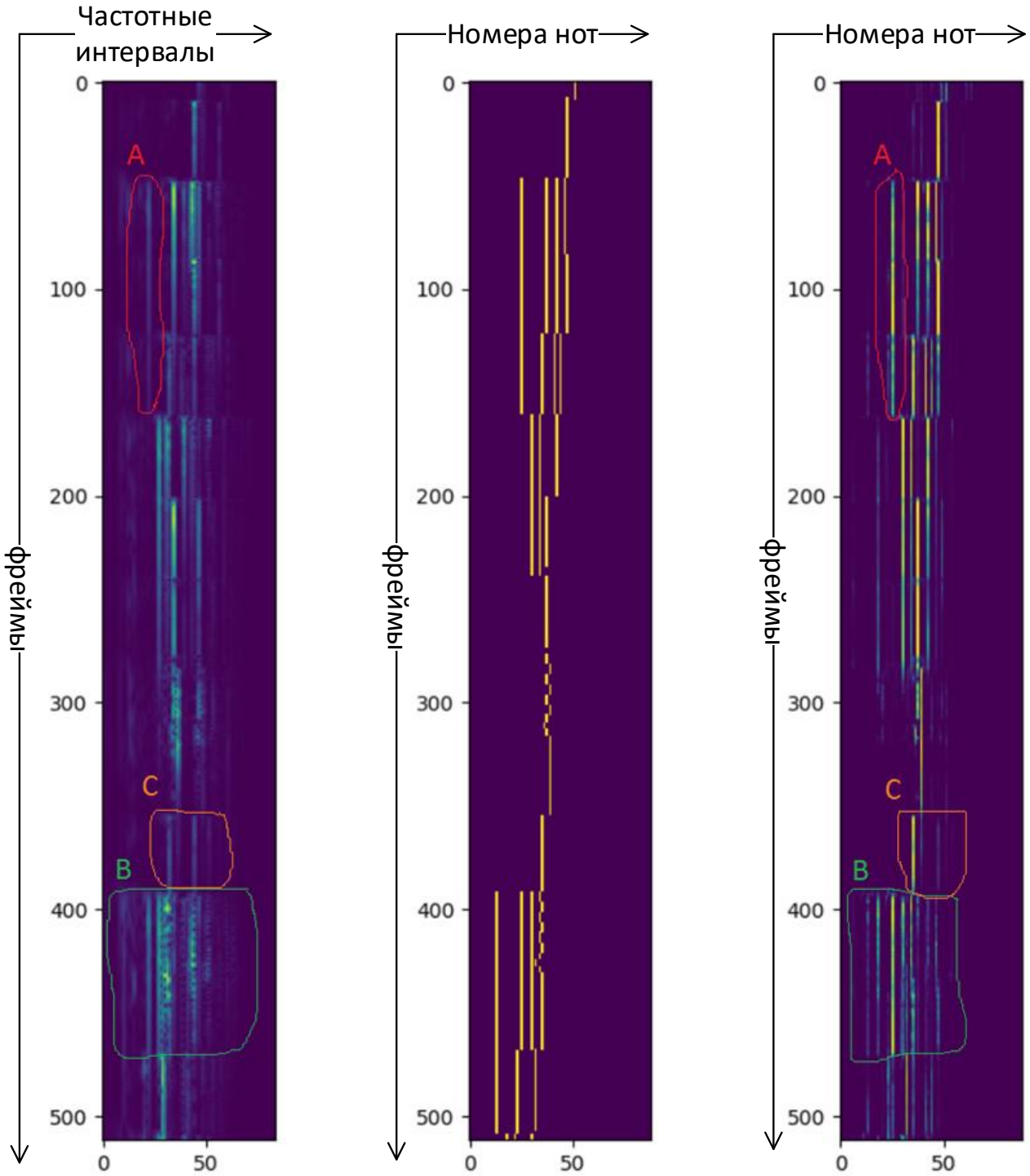

Рисунок 3.16 – Результаты классификации тестового примера №3

Тестовый пример № 3 распознан достаточно хорошо. Можно выделить несколько особенно примечательных областей на рисунке 3.16. На области A видно, что нейросеть распознала тихо сыгранную ноту и правильно предсказала ее продолжительность. Участок B на спектрограмме очень зашумлен, нейросеть все же выделила отдельные ноты, хоть и не все правильно. На участке C видно, что нейросеть угадала правильно ноту и подавила кратные гармоники.

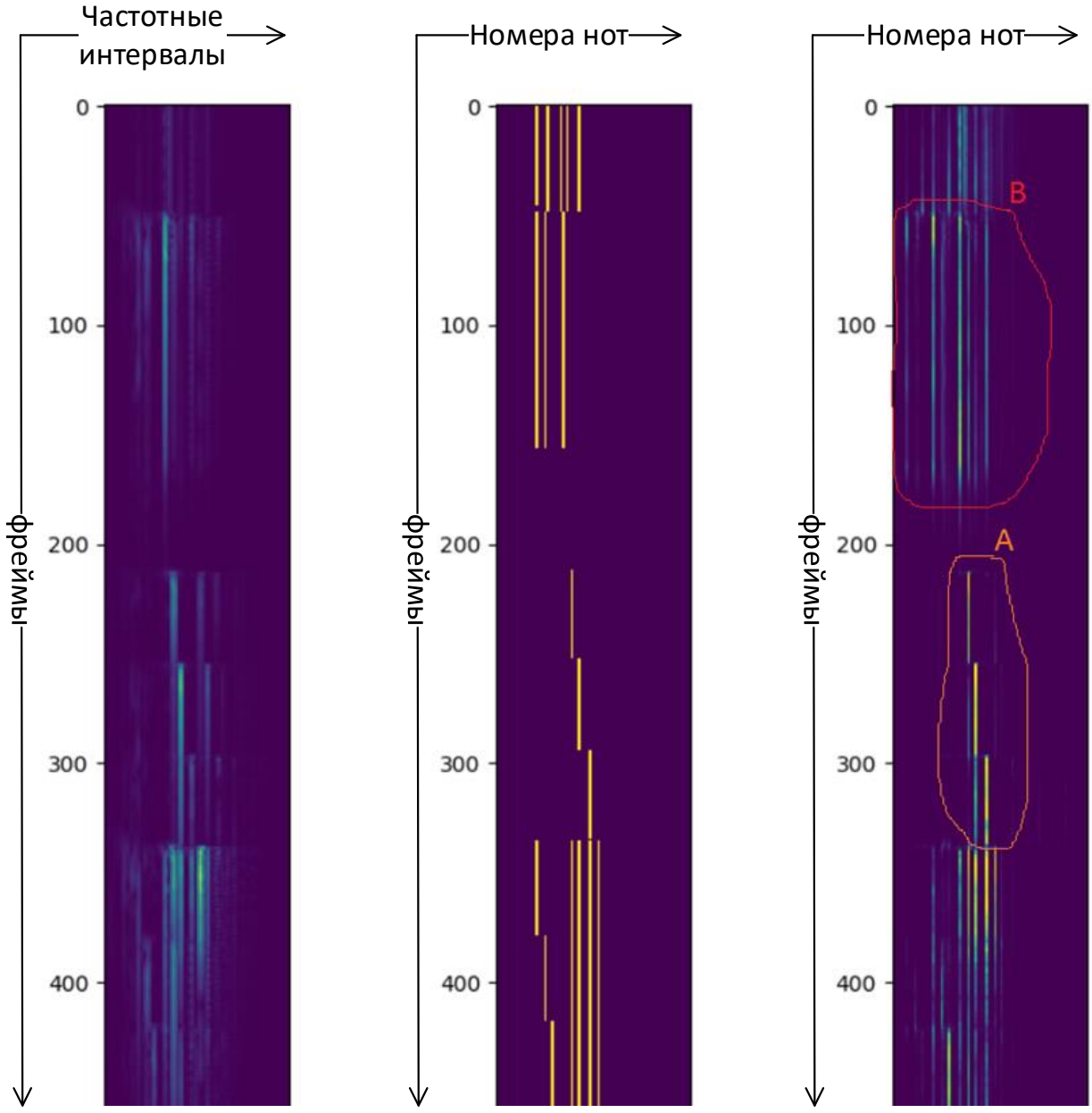

Рисунок 3.17 – Результаты классификации тестового примера №4

На рисунке 3.17 видно, что сеть успешно распознала отдельные громкие ноты в области A и подавила кратные гармоники. В области B аккорд распознан плохо, скорее всего из-за плохого качества временно-частотной характеристики.

ЗАКЛЮЧЕНИЕ

В результате исследования был реализован классификатор нот фортепиано. Для распознавания использована рекуррентная нейронная сеть типа LSTM, с помощью которой решаются специализированные задачи, в которых необходимо распознавать последовательности.

Сеть дала неплохие результаты классификации. Хотя ее уровень не позволяет сказать, что задача распознавания нот решена в общем виде. LSTM слой дал хорошие результаты по временной составляющей классификации. Длительности нот распознавались достаточно хорошо. Аналоги как раз плохо работают с длительностями нот, для улучшение результата они используют дополнительные шаги, такие как скрытые марковские модели. Глубокое обучение же позволяет не беспокоится о выборе свойств для классификации и нейросеть таким образом решает задачу в общем виде, выделяя значащие признаки самостоятельно.

В качестве будущих улучшений сети можно рассмотреть следующее:

3. Применить в архитектуре вместо слоя LSTM, слой ConvLSTM2D так как данные для обучения представлены в виде двумерного массива и имеют пространственно-временной характер.
4. Использовать новый намного больший датасет - Maestro [31]. Датасет состоит из более чем 200 часов фортепианных выступлений, записанных с точным выравниванием (~ 3 мс) между метками нот и звуковыми сигналами.
5. Улучшение характеристик частотно-временного преобразования, например, увеличение числа частотных интервалов для обучающей выборки, что позволит получить более «четкое» изображение.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Библиографический список

[1] Poliner E. A Discriminative Model for Polyphonic Piano Transcription / E. Poliner, P. W. Ellis — 2006.

[2] Нижибицкий Е. Музыкальная транскрипция при помощи методов машинного обучения — 2014.

[3] Ryyn ̈anen M. Polyphonic Music Transcription Using Note Event Modeling for MIREX / M. Ryyn ̈anen, A. Klapuri — 2008.

[4] Marolt M. A Connectionist Approach to Automatic Transcription of Polyphonic Piano Music — 2004.

[5] Zalani A. Polyphonic Music Transcription: A Deep Learning Approach / A. Zalani, A. Mittal — 2014.

[6] Звук и его свойства [Электронный ресурс]. – Электронные данные. – Режим доступа: http://theory.solfa.ru/20-вопросов-по-теории-музыки-классическ/1-звук-и-его-свойства/.

[7] Глазырин Н. — Алгоритмическое распознавание аккордов в цифровом звуке, 2014.

[8] Sleep J. — Automatic Music Transcription with Convolutional Neural Networks using Intuitive Filter Shapes, October 2017.

[9] Eyuboglu S. Note Recognition for Automatic Musical Transcription – 2014.

[10] Anssi P. Automatic Music Transcription as We Know it Today – 2014.

[11] Shevchenko N. Automatic chord transcription from digital audio – Saint-Petersburg 2015.

[12] Line E. Time-Frequency Distribution of Music based on Sparse Wavelet Packet Representations – Denmark, 2014.

[13] Tzanetakis G. Audio Analysis using the Discrete Wavelet Transform / G. Tzanetakis, G. Essl, P. Cook – Princeton, 2010.

[14] Sequence Classification with LSTM Recurrent Neural Networks in Python with Keras [Электронный ресурс]. – Электронные данные. – Режим доступа: https://machinelearningmastery.com/sequence-classification-lstm-recurrent-neural-networks-python-keras/.

[15] Zhiyao D. Automatic Music Transcription – Malaga, Spain October 26, 2015.

[16] Music Transcription using a Convolutional Neural Network

[Электронный ресурс]. – Электронные данные. – Режим доступа: https://medium.com/@dhruvverma/music-transcription-using-a-convolutional-neural-network-b115968829f4.

[17] Pesek M. Robust Real-Time Music Transcription with a Compositional Hierarchical Model / M. Pesek, Leonardis A. – University of Ljubljana, January, 2017.

[18] Sigtia S. A hybrid recurrent neural network for music transcription – Centre for Digital Music, EECS, Queen Mary University of London, 2015.

[19] Luoqi L. Music Transcription Using Deep Learning / L. Luoqi, I. Ni, L. Yang – 2014.

[20] Taegyun K. Audio-To-Score Alignment Of Piano Music Using RNN-Based Automatic Music Transcription – July 2017.

[21] Carretero M. Automatic Music Transcription Using Neural Networks – Alicante University, 2018.

[22] Sigtia S. An End-to-End Neural Network for Polyphonic Piano Music Transcription – 2016.

[23] In-depth Summary of Facebook AI's Music Translation Model [Электронный ресурс]. – Электронные данные. – Режим доступа: https://towardsdatascience.com/in-depth-summary-of-facebook-ais-music-translation-model-7516a0c3f2ce.

[24] Hawthorne C. Onsets and frames: dual-objective piano transcription / C. Hawthorne, E. Elsen – 2017.

[25] Owers J. An Exploration into the Application of Convolutional Neural Networks for Instrument Classification on Monophonic Note Samples – School of Informatics University of Edinburgh, 2016.

[26] Carthen C. Rewind: A Music Transcription Method – University of Nevada, Reno, May, 2016.

[27] Jug J. Piano transcription with deep learning – Ljubljana, 2015.

[28] Lerch A. Instrument activity detection in polyphonic music using deep neural networks – Emeryville, USA, 2016.

[29] How to Use the TimeDistributed Layer for Long Short-Term Memory Networks in Python [Электронный ресурс]. – Электронные данные. – Режим доступа: https://machinelearningmastery.com/timedistributed-layer-for-long-short-term-memory-networks-in-python/.

[30] Реализация распознавателя нот [Электронный ресурс]. – Электронные данные. – Режим доступа: https://github.com/Alexander-Andrade/piano-transcriber.

[31] Classical piano dataset [Электронный ресурс]. – Электронные данные. – Режим доступа: http://www.piano-midi.de.

[32] Maestro dataset [Электронный ресурс]. – Электронные данные. – Режим доступа: https://magenta.tensorflow.org/datasets/maestro.

Список публикаций автора

[1 – А.] Андрадэ, А. И. Музыкальная транскрипция при помощи методов машинного обучения / А. И. Андрадэ // Компьютерные системы и сети: материалы 54-й научной конференции аспирантов, магистрантов и студентов, Минск, 23 – 27 апреля 2018 г. / Белорусский государственный университет информатики и радиоэлектроники. – Минск, 2018. – С. 15.

[2 – А.] Андрадэ, А. И. Средство музыкальной транскрипции при помощи методов машинного обучения / А. И. Андрадэ, Е. В. Насуро // BIG DATA and Advanced Analytics = BIG DATA и анализ высокого уровня : сборник материалов V Международной научно-практической конференции, Минск, 13–14 марта 2019 г. В 2 ч. Ч. 1 / Белорусский государственный университет информатики и радиоэлектроники; редкол. : В. А. Богуш [и др.]. – Минск, 2019. – С. 376 – 380.

ПРИЛОЖЕНИЕ А

«Система распознавания музыкальной транскрипции при помощи методов машинного обучения». Спецификация

ПРИЛОЖЕНИЕ Б

«Система распознавания музыкальной транскрипции при помощи методов машинного обучения». Презентация