МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «МОСКОВСКИЙ ИНСТИТУТ ЭЛЕКТРОННОЙ ТЕХНИКИ»

Факультет микроприборов и технической кибернетики

Кафедра высшей математики №1

Выпускная квалификационная работа

по направлению 01.04.04 «Прикладная математика»

по направленности (профилю) - «Цифровая обработка сигналов и изображений»

на тему:

«Гармонический анализ звуковых данных с помощью глубоких нейронных сетей»

Студент группы ПМ-21М _____ Белаш М.В.

Научный руководитель, _____ Шаронов И.О.

к.т.н., старший преподаватель

Москва 2022

АННОТАЦИЯ

Выпускная квалификационная работа (ВКР) посвящена гармоническому анализу звуковых данных с помощью глубоких нейронных сетей.

Объектом исследования является процесс извлечения основных гармоник путем предварительной обработки звуковых данных (мелодии) и построения нейронной сети.

Выпускная квалификационная работа состоит из введения, трех глав, заключения и списка используемой литературы.

Введение повествует о краткой истории развития музыкального анализа, актуальности данного исследования, объекте и цели работы.

В первой главе производится обзор существующих методов классификации изображений, выводятся основные тенденции современных нейросетей.

Вторая глава посвящена изложению исследований получения основных гармоник аудиозаписей за счет процесса предобработки и построения нейронной сети.

В третьей главе полученные во второй главе методы применяются на реальных данных и, таким образом, приводятся результаты исследований.

Заключение обобщает проделанную работу и подводит краткий итог изложенным исследованиям.

Объем дипломной работы … страниц, в том числе … изображений и 15 источников литературы.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Со времен античности размышления о музыке, ее психологической и духовной силе вызывали непрекращающийся интерес, что побуждал изучать само понятие звука как природного явления, с которым человек имеет возможность работать в творческом направлении.

Материал, касающийся теории музыки, был обширен несмотря на то, что никакой систематизации способов анализа до XIX века не существовало в принципе. Еще в XVII веке Рене Декарт затрагивал теорию классической музыки, понимая слух в терминах аналитики. Для него слушатель подразумевался как субъект, и такая концепция объяснялась следующим образом: «Один и тот же мотив, от звуков которого одним хочется тотчас пуститься в пляс, у других может вызвать слезы. Ибо все здесь зависит от того, какое представление оживляет он в нашей памяти» [10]. Таким образом, благодаря Декарту появилась первая рациональная теория слушания: она отделяла воспринимающего субъекта от воспринимаемого звука, давая доступ к познанию музыки, и использовала понимание звуковых пропорций как речевых знаков.

Сменяются эпохи, направления и стили музыки, однако основные типы анализа не только не забываются, но и также являются базой для реализации новых перспективных идей, вдохновленных веяниями времени.

В наше время можно увидеть широкое распространение аудиоинформации на различных носителях: это и радио, и сеть Интернет, и компакт-диски, и многое другое. Набирают популярность различные стриминговые платформы, которые дают возможность прослушивания музыкальных каталогов по подписке, например, Spotify, VK combo и Яндекс.Музыка. В связи с этим возникает множество направлений обработки музыкальных данных, одним из которых является их поиск по фрагменту. Самым известным приложением, позволяющим решать эту проблему, является “Shazam”, с помощью которого миллионы людей по всему миру могут найти нужную им песню лишь по ее отрывку. Принцип его работы заключается в следующем: хеши, извлекаемые из записанного трека, сравниваются с имеющимися хешами из базы данных с последующими уточнениями. Однако, в том же “Shazam” можно столкнуться с проблемой сопоставления оригинальной записи и, например, обычного акапельного исполнения, то есть пения без инструментального сопровождения. Отсюда может

появиться задача создания аудио-поисковой системы, то есть системы поиска музыки по фрагменту главной мелодии, напетой или сыгранной на каком-либо инструменте. Одним из способов решения этого вопроса является распознавание нот.

Существует множество различных вариантов оценивания музыкальных композиций. Наиболее техническим способом является гармонический анализ, который помогает понять инструменты и методы, используемые композитором при написании мелодии. Человеческая слуховая система обладает очень важным свойством: она способна определять высоту звука. Благодаря этому человек может классифицировать звуки и выделять их в окружающем его пространстве, а также воспринимать музыку интонационно, то есть понимать мелодию и гармонию. Современный мир позволяет автоматизировать все эти процессы, к тому же сделать их быстрыми и точными. Так, задача распознавания нот может быть решена комбинацией из гармонического анализа и нейронных сетей за счет знания законов музыки и высоких вычислительных мощностей. Достаточно распознать главную мелодию, чтобы, например, имея банк записей нот различных музыкальных инструментов, суметь заменить звучащий инструмент или голос на необходимый.

В настоящей работе я исследую распознавание нот мелодии путем извлечения основных гармоник. Главный результат представленного исследования заключается в разработке метода, позволяющего это осуществить путем некоторой предварительной обработки звуковых данных и построения нейронной сети на основе результатов этой предобработки. Под мелодией я буду подразумевать одноголосную аудиозапись акапельного пения или сольное инструментальное исполнение без использования аккордов.

ГЛАВА 1. ОБЗОР СУЩЕСТВУЮЩИХ МЕТОДОВ

Одноголосная мелодия представляет собой некоторую последовательность звуков – нот, выстроенную таким образом, чтобы в итоге получилось некое гармоничное, благозвучное музыкальное единство. Обычно мелодия может лежать в основе некоторого большого музыкального произведения, будь то пьеса, инвенция или соната. С более научной точки зрения можно сказать, что мелодия – это сложный периодический звуковой сигнал, состоящий из последовательно идущих частот. Каждой ноте сопоставляется частота, при этом диапазон этих частот ранжируется примерно от 20 Гц до 5,3 кГц. Такой участок соответствует семи полным октавам и двум неполным. Задача распознавания нот, по сути своей, заключается в восстановлении связи нот и частот в мелодии, для которой совершается распознавание. В более упрощенном случае распознавание сводится к одной октаве, таким образом, образуя некие 12 меток, соответствующие двенадцати основным полутонам: A (ля), G# (соль-диез), G (соль), F# (фа-диез), F (фа), E (ми), D# (ре-диез), D (ре), C# (до-диез), C (до), B (си), A# (ля-диез). Отсюда вполне логично вытекает задача классификации частот, где классами будут являться указанные 12 нот.

Суть классификации как задачи распознавания строится на том, чтобы распределить по заданным образам $\mathbf{x}$ соответствующие объекты на классы C_k, где $k = 1, 2, \dots, q$. При этом классы не должны пересекаться, то есть каждый образ должен быть отнесен только к одному единственному подходящему ему классу. В случае бинарной классификации целевая переменная кодировала бы принадлежность к одному из двух классов. При многоклассовой классификации для линейных моделей осуществляется переход к набору бинарных классификаций. Тем не менее, идеей линейной классификации является возможность разделения пространства признаков на два и большее количество непересекающихся полупространств гиперплоскостями и прогнозирования в каждом из них одного из значений целевого класса.

По сравнению с классификацией, регрессия предсказывает некоторое непрерывное значение, при этом оно может быть как целым, так и с плавающей запятой. Логистическая регрессия, в отличие от обычной линейной, оценивает, к какой категории данных принадлежит конкретная точка, путем вычисления вероятности принадлежности к классу. Именно прогнозирование вероятности в обширном наборе

задач является важным требованием. Таким образом, логистическая регрессия – это частный случай линейного классификатора, который очень удобно будет использоваться и в задаче распознавания нот. Для мультиклассовой задачи по количеству классов строятся линейные модели

$$b_k(\mathbf{x}) = \langle \mathbf{w}_k, \mathbf{x} \rangle + \mathrm{w}_{0k}, \tag{1.1}$$

где $b_k(\mathbf{x})$ – оценка принадлежности для k-го класса, $\mathbf{w}_k$ – весовой вектор, w_{0k} – смещение, $\mathbf{x}$ – вектор признаков. Каждая из этих моделей дает оценку принадлежности к конкретному классу. Далее определим оператор *softmax*. Его вид выражается следующей формулой:

$$\sigma(y_i) = \frac{\exp(y_i)}{\sum_{j=1}^{n} \exp(y_j)}, \tag{1.2}$$

где n – количество классов, $\mathbf{y}$ – вектор оценок. С помощью оператора softmax оценки принадлежности классам преобразовываются в вероятности путем нормировки вектора оценок. Вероятность k-го класса будет тогда вычисляться с помощью выражения

$$P(y = k|\mathbf{x}, \mathbf{w}) = \frac{\exp(\langle \mathbf{w}_k, \mathbf{x} \rangle + \mathrm{w}_{0k})}{\sum_{j=1}^{K} \exp(\langle \mathbf{w}_j, \mathbf{x} \rangle + \mathrm{w}_{0j})}. \tag{1.3}$$

На основании указанного выше можно пробовать определяться с архитектурой нейронной сети, которая помогла бы разрешить стоящую в данном исследовании проблему. На сегодняшний день существует множество различных архитектур сетей для решения самых разнообразных задач. Однако среди них можно выделить наиболее часто применяемые моменты. Так, например, в большинстве своем самыми используемыми слоями в таких сетях являются *сверточные* и различные вариации слоев *подвыборки* (*пулинга*), будь то перекрывающий пулинг, макспулинг или средний пулинг. Сверточные слои осуществляют операцию дискретной свертки для формирования выходной карты признаков (feature map), чтобы отфильтровать маловажные детали и выделить существенное. Пусть входное изображение обозначается как $\mathbf{a} = (a_{j,k})$. Тогда для каждой m-ой карты признаков $\mathbf{f}^{(m)} = (f_{x,y}^{(m)})$ будет верна следующая формула:

$$f_{x,y}^{(m)} = h\left(\left(\mathbf{w}^{(m)} * \mathbf{a}\right)_{x,y} + b^{(m)}\right) = h\left(\sum_j \sum_k w_{j,k}^{(m)} a_{x-j,y-k} + b^{(m)}\right), \tag{1.4}$$

где $\mathbf{w}^{(m)} = \left(w_{j,k}^{(m)}\right)$ – ядро двумерной свертки, $b^{(m)}$ - смещение. Слои пулинга применяются для уменьшения размерности входного тензора, что позволяет сократить количество дальнейших вычислений, а также добавляет устойчивость к небольшим сдвигам объектов на изображении. Можно было заметить, что указанные слои применяются для обработки изображений. Как будет показано в следующей главе, несмотря на то, что первично в задаче распознавания нот начальными данными являются звуковые файлы, нейронные сети будут принимать на вход именно изображения. Для устранения переобучения чаще всего применяется *метод прореживания*, т.е. слой *Dropout*. Смысл метода - исключить на каждой итерации изменения весовых коэффициентов часть весов с некоторой заданной вероятностью. На выходе ставится слой softmax, который задается с помощью формулы (1.2).

Для примера рассмотрим сеть MobileNet [13, 14]. MobileNet — это модернизированная архитектура, которая использует разделяемые по глубине свертки для создания облегченных глубоких сверточных нейронных сетей и обеспечивает эффективную модель для мобильных и встроенных приложений машинного зрения. Это первая мобильная модель компьютерного зрения TensorFlow. Схема и архитектура сети представлены на рисунках 1.1 и 1.2 соответственно. Как можно заметить, помимо множества сверточных слоев сеть включает в себя также слой среднего пулинга, то есть размерность пространства карт признаков уменьшается за счет усреднения, а также полносвязный слой, за которым следует выходной слой softmax. Остановимся поподробнее на сверточных слоях. MobileNet использует отделимые по глубине свертки, что значительно уменьшает количество параметров по сравнению с сетью с обычными свертками с одинаковой глубиной в сетях. Это приводит к созданию легких глубоких нейронных сетей. Отделимая по глубине свертка производится из двух операций: глубинной свертки и точечной свертки. Фильтр глубинной свертки выполняет одну свертку для каждого входного канала, а фильтр точечной свертки объединяет выходные данные глубинной свертки линейно со сверткой 1*1, как показано на рисунке 1.3. MobileNets разделяют свертку на глубинные свертки 3*3 и точечную свертку 1*1, что демонстрируется на рисунке 1.4. Множитель ширины α, отвечающий за количество каналов в каждом слое и множитель глубины (разрешения) ρ, который ответственен за пространственные размеры входных тензоров, будут являться гиперпараметрами архитектуры.

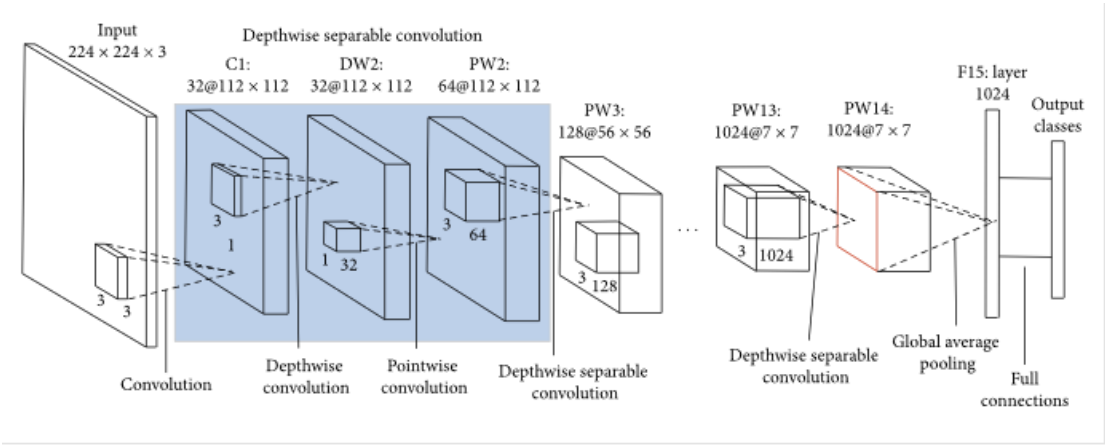


Рисунок 1.1. Схема архитектуры сети MobileNet

Type / Stride	Filter Shape	Input Size
Conv / s2	3 × 3 × 3 × 32	224 × 224 × 3
Conv dw / s1	3 × 3 × 32 dw	112 × 112 × 32
Conv / s1	1 × 1 × 32 × 64	112 × 112 × 32
Conv dw / s2	3 × 3 × 64 dw	112 × 112 × 64
Conv / s1	1 × 1 × 64 × 128	56 × 56 × 64
Conv dw / s1	3 × 3 × 128 dw	56 × 56 × 128
Conv / s1	1 × 1 × 128 × 128	56 × 56 × 128
Conv dw / s2	3 × 3 × 128 dw	56 × 56 × 128
Conv / s1	1 × 1 × 128 × 256	28 × 28 × 128
Conv dw / s1	3 × 3 × 256 dw	28 × 28 × 256
Conv / s1	1 × 1 × 256 × 256	28 × 28 × 256
Conv dw / s2	3 × 3 × 256 dw	28 × 28 × 256
Conv / s1	1 × 1 × 256 × 512	14 × 14 × 256
5× Conv dw / s1	3 × 3 × 512 dw	14 × 14 × 512
5× Conv / s1	1 × 1 × 512 × 512	14 × 14 × 512
Conv dw / s2	3 × 3 × 512 dw	14 × 14 × 512
Conv / s1	1 × 1 × 512 × 1024	7 × 7 × 512
Conv dw / s2	3 × 3 × 1024 dw	7 × 7 × 1024
Conv / s1	1 × 1 × 1024 × 1024	7 × 7 × 1024
Avg Pool / s1	Pool 7 × 7	7 × 7 × 1024
FC / s1	1024 × 1000	1 × 1 × 1024
Softmax / s1	Classifier	1 × 1 × 1000

Рисунок 1.2. Архитектура MobileNet (версия 1)

MobileNets — это семейство мобильных моделей компьютерного зрения TensorFlow, предназначенных для эффективного повышения точности с учетом ограниченных ресурсов для приложений. Это небольшие модели с малой задержкой и низким энергопотреблением, параметризованные для соответствия ограничениям ресурсов для различных вариантов использования. Классификация, обнаружение, встраивание и сегментация – задачи, где они применяются.

Также можно вспомнить довольно-таки уже старую нейросеть LeNet5 [12], состоящую из 7 слоев, в дополнение к входному, при этом 3 слоя этой сети – сверточные, 2 слоя – для подвыборки, а на выходе снова стоит слой softmax. Еще один

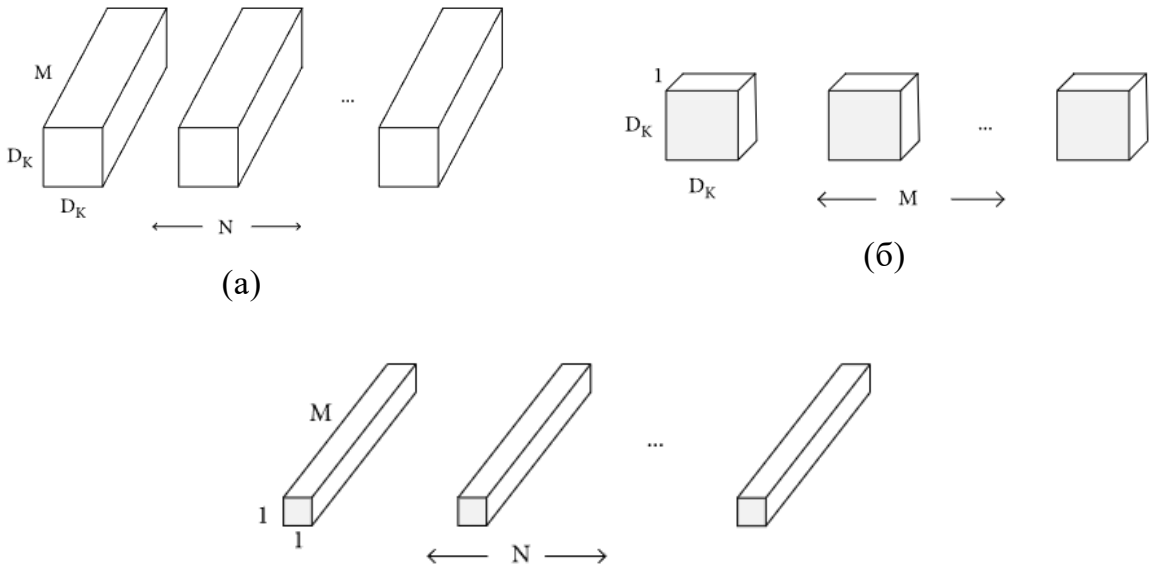
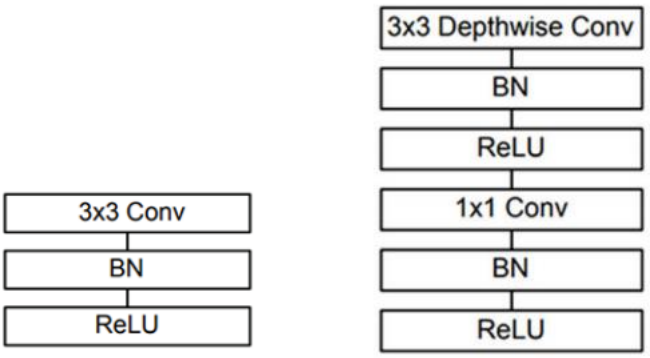


Рисунок 1.3. Стандартные сверточные фильтры и фильтры с разделением по глубине. (а) - стандартные сверточные фильтры, (б) - глубинные сверточные фильтры, (в) - точечные сверточные фильтры.



(а) (б)

Рисунок 1.4. (а) Стандартный сверточный слой с пакетной нормализацией и ReLU. (б) Разделимая по глубине свертка с глубинными и точечным слоями с последующей пакетной нормализацией и ReLU.

пример – сеть GoogLeNet [15]. Не заостряя внимание на особенный для этой нейросети слой inception, стоит сказать, что в ней используются не только сверточные слои и выходной softmax-слой, но теперь еще и две вариации пулинга: максимальный и средний. Также для устранения переобучения здесь применяется слой Dropout с выбросом 40% весов.

Как можно заметить, сверточные нейронные сети пользуются в настоящее время большой популярностью, несмотря на их недостатки, такие как продолжительное

время обучения и большая вероятность переобучения при недостаточном количестве данных для тренировки. В своем же исследовании я постараюсь отойти от этой тенденции и построить более простую сеть. Тем не менее, основная сложность будет заключаться в предварительной обработке имеющихся данных, но об этом будет рассказано уже в следующей главе.

ГЛАВА 2. ИССЛЕДОВАНИЕ ПОЛУЧЕНИЯ ОСНОВНОЙ ГАРМОНИКИ

В настоящей главе приводятся основные моменты исследования получения основной гармоники из аудиозаписи. Всю исследовательскую работу, как было указано ранее, можно разбить на два этапа: предобработку данных и построение нейронной сети. Основные модификации, производимые для улучшения результатов обучения нейросети, будут произведены на этапе предварительной обработки, поскольку наше внимание в данной работе сосредотачивается именно на гармоническом анализе звуковых данных.

2.1. Предобработка

Важное место в получении модели нейронной сети, с помощью которой мы сможем распознавать ноты, занимает обработка данных. Для любой сети предобработка помогает увеличить скорость поиска оптимальных параметров и, кроме того, сделать результаты более точными. Для использования предварительной обработки в данном исследовании существует еще одна немаловажная причина. Заключается она в том, что с ее помощью можно свести обработку звуковых сигналов непосредственно к работе с изображением. Именно из изображения в этом исследовании я буду извлекать признаки и на их основе обучать нейронную сеть.

2.1.1. Подготовка базы данных для распознавания нот

Для того, чтобы обучить нейронную сеть распознавать ноты в произведениях и протестировать ее необходимо сначала найти базы данных. В рассматриваемом случае данные представляют из себя аудиофайлы формата WAV, являющиеся моно-сигналами, при этом для каждого файла должна иметься следующая информация:

- Частота в герцах или высота тона;
- Время появления каждой ноты в мелодии в секундах;
- Время окончания или длительность каждой ноты в секундах.

В случае, когда в базе даны длительности нот, их можно легко конвертировать во время окончания с помощью формулы

$$t_{end} = t_{start} + dt, \tag{2.1}$$

где t_{start} – время начала, dt – длительность, t_{end} – искомое время окончания ноты.

Слуховая система человека может различить высоту тона только тогда, когда звуковой сигнал является периодическим. При простом гармоническом колебании, таком как, например, синусоидальный сигнал, период колебаний T будет определять частоту $f = 1/T$, из чего следует, что для различия высоты частота звукового сигнала будет являться определяющим параметром. В случае же более сложного звука слуховая система определяет высоту по основному тону, однако периодичность сигнала все еще является важным фактором. Спектр должен состоять из гармоник, иными словами, обертонов, частоты которых относятся друг к другу целочисленно. В иных ситуациях, при невыполнении этого условия высота тона не может быть определена слуховой системой [7].

Для того, чтобы в дальнейшем получить информацию о нотах, удобнее будет работать с частотой. Поэтому если вместо нее предоставлены данные о высоте тона ноты, то для перехода к частотам будем использовать стандарт midi

$$f = 440 \cdot 2^{\frac{p-69}{12}}, \tag{2.2}$$

где f – частота в Гц, p – высота тона.

Также, опять же для удобства, переведем секунды в отсчеты. Частота дискретизации мелодий стандартно рассматривается равной 44,1 кГц, то есть 44100 отсчета в секунду. Значит, для перевода времени в отсчеты воспользуемся формулой

$$n = round(t) \cdot 44100, \tag{2.3}$$

где n – количество отсчетов, t – время в секундах, $round()$ – операция округления.

В рамках проделанной работы рассматривалось 2 набора данных: «База данных с маркировкой звука флейты для автоматической транскрипции музыки» [2] - набор, который, как понятно из названия, состоит из сольных мелодий, исполненных на флейте, и «CSD: набор данных детских песен для исследования певческого голоса» [3], который содержит аудиозаписи акапельного одноголосного пения. При этом вторая база также разделялась на два подкаталога: английских и корейских песен. Обработка этих наборов состоит в следующем:

1) Сначала происходит загрузка баз данных и меток;

2) Затем осуществляется дополнительное разбиение загруженных данных на маленькие интервалы, пригодные для обучения;

3) В конце совершается также дополнительное разбиение меток согласно данным.

Получившиеся фрагменты необходимо распределить на каталоги, соответствующие конкретным нотам. Такое распределение осуществимо из соответствия нот и частот. В этом исследовании я не буду опираться на различие октав, поэтому основываюсь на 12 основных полутонах. Частоты всего звукоряда можно получить из соответствующих таблиц, однако также существует математическая формула, с помощью которой есть возможность их подсчитать. Эта формула имеет вид

$$f(i) = f_0 \cdot 2^{i/12}, \tag{2.4}$$

где f_0 – частота камертона, то есть частота эталонного звука (обычно соответствует ноте А первой октавы и равна 440 Гц), i – количество полутонов в интервале от исследуемого звука к эталону f_0.

Из формулы (2.4) можно вывести количество полутонов i, на которое заданная частота отстает от частоты камертона. Выражение получается следующим:

$$i = \frac{\ln \frac{f_0}{f(i)}}{\ln 2^{\frac{1}{12}}}. \tag{2.5}$$

Как уже было сказано выше, различие октав в данном исследовании учитывать не будет. Поэтому для сведения всех нот к 12 основным полутонам найдем значение z из следующей системы:

$$z = \begin{cases} round(i)\%12, & z \geq 0, \\ 12 + round(i)\%12, & z < 0, \end{cases} \tag{2.6}$$

где % - операция получения остатка от деления. При этом, если $z < 0$, то разворачиваем ноту так, чтобы она оказалась выше ноты А. Это отображается во второй строке системы. Нота, соответствующая заданной частоте без учета октав, определяется из значения z по таблице 2.1.

Таким образом, теперь мы имеем 12 каталогов, каждый из которых соответствует одной из 12 нот и содержит различные короткие звукозаписи этой ноты.

2.1.2. Генерация спектрограмм

Теперь необходимо определиться, какие данные будут использоваться в нейронной сети как входные. Ранее уже упоминалось, вместо работы со звуком я буду

z	Нота
0	A
1	G#
2	G
3	F#
4	F
5	E
6	D#
7	D
8	C#
9	C
10	B
11	A#
6	D#

Таблица 2.1. Соответствие нот и их номеров

использовать изображения как входные данные нейросети. Часто в качестве такого изображения, которое, можно сказать, отображает звук, используют спектрограммы. *Спектрограмма* – это способ демонстрации уровня сигнала или «громкости» сигнала на различных частотах в течение времени в виде изображения. Они используются в задачах, где необходимо отобразить частоты записанных или сгенерированных звуковых волн. В сейсмологии, например, спектрограммы применяются для визуализации частоты непрерывных сигналов, которые регистрируются одним или несколькими сейсмометрами с целью различия и описания ряда видов сейсмической активности, таких как землетрясения и другие вибрации, наблюдаемые в земле. Спектрограмма представляет собой матрицу размера $N \times M$, где N соответствует определенной частоте, а каждый столбец, которых всего M штук, является спектром фрагмента аудиозаписи. На рисунке 2.1 показан пример спектрограммы. Как из него видно, чаще всего спектрограммы представляются в виде 2D-графика: вдоль горизонтальной оси строится время, вдоль вертикальной - частота. Яркость или цвет точек изображения представляет силу частотной составляющей в каждом временном

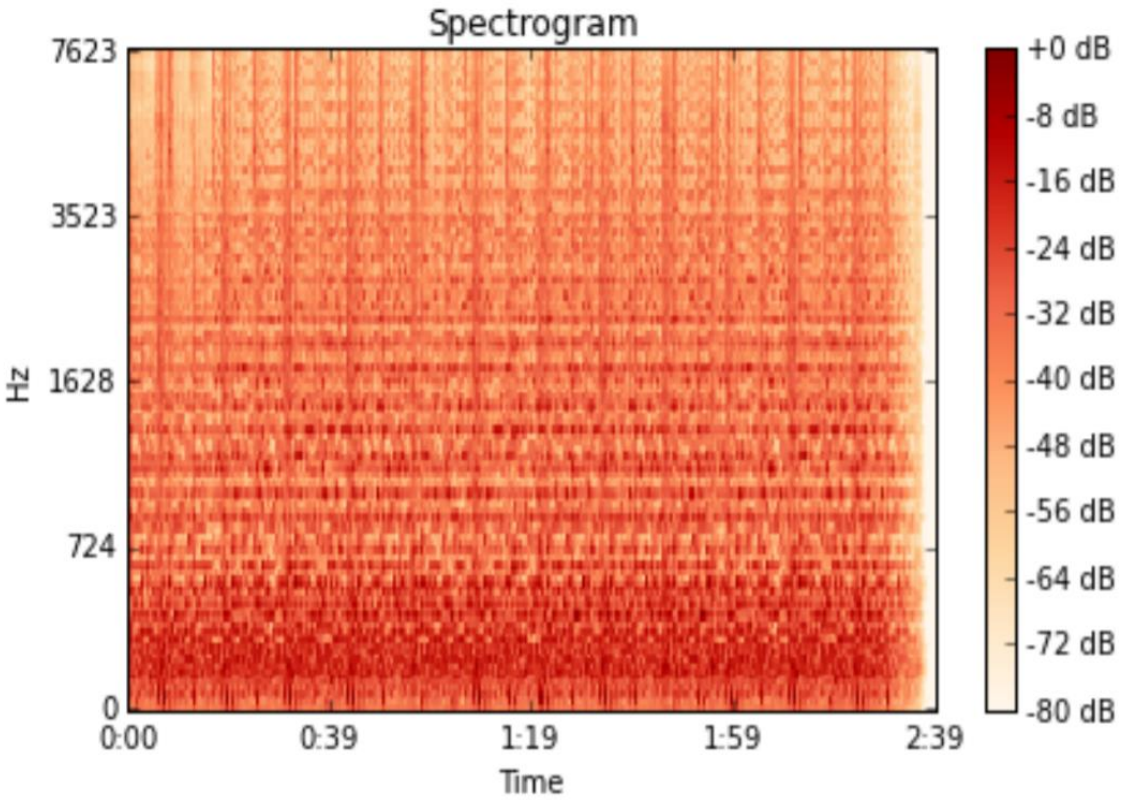


Рисунок 2.1. Пример спектрограммы (Rick Astley - Never Gonna Give You Up)

интервале и откладывается в третьем измерении. Также оси абсцисс и ординат могут быть переставлены местами, а амплитуда иногда представлена не цветом и интенсивностью, а вершинами в трехмерном пространстве. В зависимости от цели использования спектрограммы частоту и амплитуду, отложенные на осях, рассматривают как линейными, так и логарифмическими. Обычно аудио представляется с помощью логарифмической оси амплитуды, а единицей измерения являются децибелы (или дБ), частота же – линейная для подчеркивания гармонических отношений, а для задач выделения музыкальных отношений - логарифмическая.

Часто используют один из двух вариантов генерации спектрограммы. Ранее в условиях отсутствия необходимых ресурсов и методов цифровой обработки сигналов прибегали только к аппроксимации в качестве набора полосовых фильтров. С развитием технологий спектрограммы стали рассчитывать за счет временного сигнала с помощью *оконного преобразования Фурье*

$$X[n] = \sum_{j=0}^{J-1} w(j)x(t_j)e^{-\frac{i2\pi nj}{J}} \qquad (2.7)$$
$$n = 0, 1, \dots, N-1$$

где J – размер анализируемого фрагмента звукозаписи в отсчетах, $w(j)$ – оконная функция. Оконное преобразование Фурье также называют *коротким* (или *кратковременным*) *преобразованием Фурье* (*STFT*). Я буду применять второй способ. Поскольку спектр сигнала состоит из комплексных значений, для сопоставления его со временем используем абсолютное значение спектра (модуль), а не сам спектр.

Аудиосигнал постоянно меняется, поэтому для упрощения я предполагаю, что на коротких временных масштабах аудиосигнал, наоборот, статически не будет сильно преобразовываться (в реальности это не так). Существуют разные варианты весовых окон, которые можно применить в STFT, однако наибольшей популярностью пользуется окно Хэмминга:

$$w(n) = 0.54 - 0.46 \cdot \cos\left(\frac{2\pi n}{N-1}\right), \tag{2.8}$$

где $0 \leq n \leq N-1$, N – длина окна.

Вместо спектра самого сигнала можно использовать его мощность:

$$P = \frac{|STFT(x_i)|^2}{N}, \tag{2.9}$$

где x_i – i-ый кадр сигнала x, N – длина окна. Это мотивировано человеческой ушной улиткой, которая вибрирует в разных местах в зависимости от частоты входящих звуков. В соответствии с тем, где расположены в вибрирующей улитке различные нервы, они срабатывают и сообщают мозгу о том, что найдены определенные частоты. Оценка спектрограммы выполняет аналогичную работу для нас, определяя, какие частоты присутствуют в кадре.

Спектральная оценка спектрограммы содержит много информации, которая не требуется для автоматического распознавания нот. В частности, улитка не может выделить разницу между двумя близко расположенными частотами. Этот эффект становится более выраженным по мере увеличения частот. Поэтому я буду брать не сплошную спектрограмму, а скопления ее блоков и просуммирую их, чтобы получить представление о том, сколько энергии существует в различных частотных областях. Это выполняется с помощью банков фильтров Мела: по мере увеличения частоты фильтры расширяются, поскольку вариации вызывают меньше беспокойства. Меня будет интересовать только приблизительно, сколько энергии приходится на каждый фрагмент. Мелы являются специальными единицами, в которых откладываются значения высоты звука. Одному мелу при уровне 40 дБ будет соответствовать

ощущаемая высота звука частотой 1000 Гц (для оценки высоты тона также иногда может использоваться другая единица – барк, равный 100 мелам). Уши человека ощущают окружающие звуки нелинейно, и шкала мела как раз направлена на то, чтобы имитировать такое нелинейное такое восприятие, будучи более различимой на более низких частотах и менее различимой на более высоких частотах. Существуют различные варианты конвертации герц (f) в мелы (m) и наоборот. Один из самых распространенных способов перехода между этими величинами показан формулами (2.10) и (2.11):

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right) = 1127 \ln\left(1 + \frac{f}{700}\right), \tag{2.10}$$

$$f = 700\left(10^{\frac{m}{2595}} - 1\right) = 700\left(e^{\frac{m}{1127}} - 1\right). \tag{2.11}$$

Для получения блоков спектрограмм я буду использовать банк треугольных фильтров: каждый такой фильтр имеет треугольную форму с откликом 1 на центральной части и линейно уменьшается к 0, пока не достигнет центральных частот двух соседних фильтров, где отклик равен 0. Моделирование этих фильтров можно осуществить с помощью следующего уравнения:

$$H_m(k) = \begin{cases} 0, & k < f(m-1), \\ \dfrac{k - f(m-1)}{f(m) - f(m-1)}, & f(m-1) \le k < f(m), \\ \dfrac{f(m+1) - k}{f(m+1) - f(m)}, & f(m) \le k \le f(m+1), \\ 0, & k > f(m+1), \end{cases} \tag{2.12}$$

где f – список из $M + 1$ меловых частот. На рисунке 2.2 показан пример банка треугольных фильтров. Применять этот банк я буду путем умножения его на мощность спектра (2.9):

$$S = P \cdot H, \tag{2.13}$$

где H – банк треугольных фильтров, P – спектр мощности сигнала, «·» - операция матричного умножения.

Для оценки высоты тона в музыке могут также использоваться и другие – музыкальные – шкалы: тоны и полутоны, октавы и другие музыкальные интервалы. Тем не менее, связь с психофизической шкалой высоты тона, которая построена для чистых тонов, не является однозначной. Так, например, увеличение высоты тона на октаву до частоты примерно 5000 Гц связано с увеличением частоты в два раза

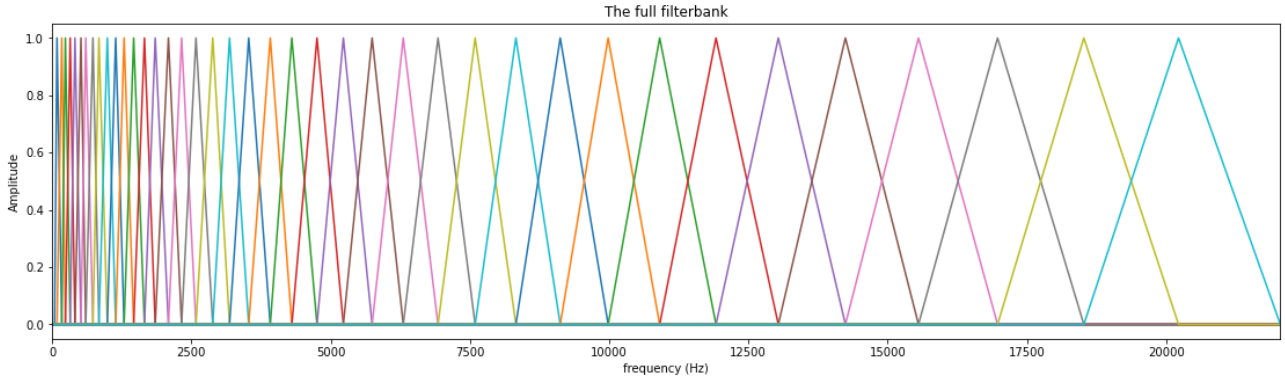


Рисунок 2.2. Пример банка треугольных фильтров

(например, ноты А первой и второй октав соответствуют частотам 440 Гц и 880 Гц соответственно). Однако, для частот выше 5000 Гц это соотношение будет нарушено: для перехода на октаву выше понадобится текущую частоту увеличить почти в 10 раз. Этот факт дал основание некоторым ученым предложить две размерности высоты тона: психофизическую, измеряемую в мелах, пропорциональную логарифму частоты в некоторых пределах и установленную для чистых тонов (pitch height), и музыкальную, которая соответствует названию нот (pitch chroma) и может быть определена примерно до 5000 Гц.

Теперь от полученных энергий фильтрующих банков необходимо найти логарифм. Это снова объясняется нелинейностью восприятия звуков человеком. На последнем шаге остается посчитать дискретное косинусное преобразование (ДКП) от полученного набора прологарифмированных энергий. Дискретное косинусное преобразование (ДКП) – одно из самых известных преобразований в математике, используемое для сжатия двумерных цифровых изображений. Это преобразование имеет быстрый алгоритм вычисления, то есть реализуется порядком $NlogN$ вычислительных операций, и показывает наибольшую эффективность для кодирования сигнала. Для матрицы X двумерное ДКП определяется следующим образом [5]:

$$y_{k,l} = \frac{2}{\sqrt{MN}} c(k)c(l) \sum_{j=0}^{N-1} \cos\left(\frac{\pi k}{N}\left(j+\frac{1}{2}\right)\right) \sum_{m=0}^{M-1} x_{j,m} \cos\left(\frac{\pi l}{M}\left(m+\frac{1}{2}\right)\right) \qquad (2.14)$$

$$k = 0, 1, \dots, N-1, \quad l = 0, 1, \dots, M-1,$$

где $X = \left(x_{j,m}\right)_{j,m=0}^{N-1,M-1}$,

$$c(k) = \begin{cases} \frac{1}{\sqrt{2}}, & k = 0, \\ 1, & k \neq 0. \end{cases}$$

Наборы фильтров, используемые для получения спектрограмм, перекрываются, и их энергии хорошо коррелируют друг с другом. ДКП показывает результаты декорреляции, наиболее близкие к преобразованию Карунена-Лоэва, которое является эталонным.

Таким образом, после всех действий получается *кепстр* или *мел-кепстральные коэффициенты MFCC* — представление кратковременного спектра мощности звука, который основан на линейном косинусном преобразовании логарифмического спектра мощности на нелинейной мел-шкале частоты. Другими словами, MFCC - особый спектр, из которого с помощью различных фильтраций и преобразований удалены незначительные для человеческого слуха компоненты. Предполагается, что на непересекающихся отрезках сигнала длиной 20-40 мс частоты не сильно меняются, то есть спектр носит кратковременный характер. Обычно коэффициенты MFCC используются как функции в системах распознавания речи: например, системы, автоматически распознающие числа, произнесенные по телефону. Звуки, которые воспроизводятся человеком, определяются формой голосового тракта, которая описывается огибающей спектра. MFCC должно как можно точнее представлять эту огибающую. Также часто MFCC применяются как раз-таки в приложения для поиска музыкальной информации: классификация жанров, меры сходства звука и тому подобное.

2.1.3. Параметры входных данных

Пусть N – количество отсчетов файла, S – шаг окна, K – размер окна, M – количество спектрограмм, подаваемых на вход.

Далее рассмотрим основные параметры, которые могут меняться в зависимости от модели.

1) Размер окна

Человеческое ухо не слышит колебания меньше 20 Гц. Для частоты дискретизации 44100 отсчетов/секунда получаем тогда размер окна 2205. Для красоты возьмем 2048 отсчетов, что составляет примерно 21.5 Гц – это и может послужить первоначальным размером фрейма для оконного преобразования Фурье.

Тем не менее, дискретное преобразование Фурье сталкивается с принципом неопределенности время-частотного разрешения, которое определяется уравнением

$\Delta\nu \cdot \Delta t = \frac{1}{K}$. Поскольку частота дискретизации равна 44100 отсчетов/секунда, $\Delta t = \frac{1}{44100}$ и является расстоянием между отсчетами во времени. Я буду рассматривать 7 октав, где минимальная частота примерно соответствует 55 Гц, а максимальная – 7040 Гц. Основные музыкальные частоты определяются с помощью выражения

$$f = 440 \cdot \left(2^{\frac{1}{12}}\right)^n, \tag{2.15}$$

где при $n = 0$ получается $f = 440$ Гц – частота камертона. Поэтому самое маленькое расстояние между частотами наблюдается в нижних частотах $\nu_1 - \nu_0 = 55 \cdot \sqrt[12]{2} - 55 = 55 \cdot \left(\sqrt[12]{2} - 1\right)$. Таким образом, общая формула расстояния между соседними нотами равна

$$\Delta\nu = 55 \cdot \left(\sqrt[12]{2}\right)^n \left(\sqrt[12]{2} - 1\right). \tag{2.16}$$

Как понятно, величина $\Delta\nu$ – неравномерная, ее минимум получается равным $\Delta\nu_{min} = \nu_1 - \nu_0 \approx 3.3$. Если округлить $\Delta\nu_{min}$ до значения 4, то $K = \frac{44100}{4} = 11025$. Таким образом, получается новый вариант окна, при котором шаг по частоте будет почти идеальным. При этом, если брать $K = 2048$, то получится шаг по частоте, равный $\frac{44100}{2048} = 21.53$ Гц, а это слишком большая величина.

2) Количество спектрограмм

После оконного преобразования Фурье получаем множество коротких спектров, длина которого вычисляется следующим образом:

$$L = N//2 + 1, \tag{2.17}$$

где L – длина множества коротких спектров, N – размер кадра, «//» – операция целочисленного деления. Множество определенного количества последовательных спектрограмм составит пачку. Тогда входом нейросети будет являться одна такая пачка. В своей диссертации [1] Корзениовски предлагал брать 15 последовательных спектрограмм. В этом исследовании я буду пытаться подобрать такое количество спектрограмм в пачке, которое даст наилучшую точность.

Всего мы имеем без ограничений N/K окон размера S. Обозначим $T = N/K$. Ограничение количества спектрограмм состоит в следующем:

- Если $T < M$, то дополняем спектрограмму последним столбцом до количества M штук (в ходе исследования также рассматривался вариант заполнения недостающих столбцов усреднением имеющихся, однако результаты получались примерно

одинаковые при больших вычислительных затратах);

- Если $T > M$, то случайно выбираем окно спектрограмм длиной M столбцов; при этом имеем $T - M + 1$ вариантов выбора этого окна;
- Если $T = M$, тогда ограничений нет.

Исходя из таких условий, получаем пакет спектрограмм размера M, который подаем на вход нейросети.

3) Шаг окна

Обычно в качестве шага окна выбирают половину его размера, как, например, в аудиокодеках. Однако можно попробовать варьировать его величину. Я буду брать такие варианты, которые не превышают размер окна.

4) Изменение треугольных фильтров

Стандартно берут 20-40 треугольных фильтров. Для начала возьмем 40. Тогда вход нейросети будет иметь размерность $40 \times M$. Для вычисления банка фильтров минимальной частотой возьмем 21.5 Гц. Поскольку сигнал действительный, верхняя частота ограничена половиной частоты дискретизации, то есть 22050 Гц, поэтому ее и возьмем за максимум. С помощью уравнения (2.10) преобразуем эти величины в мелы. Количество точек на шкале мелов, которые нам понадобятся для 40 фильтров, равно 42, что означает, что необходимо еще, помимо максимального и минимального значения, 40 дополнительных точек, которые мы расположим линейно. Далее с помощью уравнения (2.11) переводим получившиеся точки обратно в Герцы, при этом можно обратить внимание, что начальная и конечная точка находятся на тех частотах, которые были заданы изначально. Поскольку мы не имеем частотного разрешения, необходимого для размещения фильтров в точных точках, рассчитанных ранее, нужно округлить эти частоты до ближайшего бина БПФ. Этот процесс не будет влиять на точность функций. Чтобы преобразовать частоты в номера бинов БПФ, мы должны знать размер БПФ и частоту дискретизации. Тогда частоты округляются с помощью следующего выражения:

$$f(i) = floor\left(\frac{(K+1)h(i)}{44100}\right), \tag{2.18}$$

где $h(i)$ – посчитанные ранее частоты, $f(i)$ – округленные частоты, $1 \leq i \leq 42$ – номер фильтра, $floor()$ – операция округления в меньшую сторону. Далее создаем банки фильтров. Первый банк начнется в первой точке, достигнет своего пика во второй

точке, затем вернется в третью точку. Второй банк начнется со второй точки, достигнет своего максимума в третье, затем будет равен нулю в четвертой и так далее. Таким образом, для вычисления банка фильтров используем формулу (2.12).

Далее рассмотрим еще один вариант генерации треугольных фильтров. Пики фильтров должны лежат как можно ближе к основным музыкальным частотам, которые выражаются формулой (2.15). Как было сказано при определении размера окна, минимальная рассматриваемая частота равна 55 Гц, максимальная – 7040 Гц. Отсюда получается, что n необходимо рассматривать равным от -36 до 48. Поскольку крайние значения в расчет не берутся, получается 83 фильтра. Распределение точек на шкале мела берется из соответствия формуле (2.15).

2.1.4. Применение модифицированного дискретного косинусного преобразования

Представление сигнала в виде модифицированного дискретного косинусного преобразования (МДКП) стало доминирующим инструментом в высококачественном кодировании звука из-за его особых свойств. В дополнение к возможности уплотнения энергии, аналогичной ДКП, МДКП также обеспечивает критическую выборку, уменьшение эффекта блокирования и гибкое переключение окон. МДКП является Фурье – преобразованием, основанным на ДКП, и имеет следующий вид:

$$y_r = \sum_{k=0}^{2N-1} \tilde{x}_k \cos\left\{\pi \frac{\left[k + \frac{N+1}{2}\right]\left(r + \frac{1}{2}\right)}{N}\right\}, \quad r = 0, \dots, N-1, \tag{2.19}$$

где $\tilde{x}_k = h_k x_k$ – оконный входной сигнал, x_k – входной сигнал из $2N$ отсчетов, h_k – оконная функция. Вещественные числа преобразуются в вещественные, поэтому работа с комплексными числами здесь не требуется.

МДКП является преобразованием с перекрытием, то есть каждый из блоков объемных наборов данных, на которых выполняется преобразование, перекрывается, причем шаг смещения такого окна равен половине его размера.

Оконная функция имеет следующие ограничения:

$$\begin{aligned} h_k &= h_{2N-1-k}, \\ h_k^2 + h_{k+N}^2 &= 1. \end{aligned} \tag{2.20}$$

В качестве оконной функции я буду использовать дополнительное окно мощности Vorbis, выражающееся формулой

$$w(n) = \sin\left[\frac{\pi}{2}\sin^2\left(\frac{\pi\left(n+\frac{1}{2}\right)}{2L}\right)\right]. \quad (2.21)$$

Таким образом, я попробую заменить STFT на МДКП, тем самым получив еще одну модель для сравнения.

2.1.5. Итоговые модели предобработки

Исходя из всего вышесказанного, для предобработки звукового сигнала можно выделить 4 подхода, которые далее будут рассматриваться в текущем исследовании. Эти подходы изложены в таблице 2.2.

В моделях STFT и fSTFT можно заметить, что формула (2.7) применяется не в чистом виде, а фактически получается формула (2.9). Делается это с той целью, что в дальнейшей нейронной сети нормализационных слоев не предусмотрено, но при этом необходимо соблюдать равенство Парсеваля, согласно которому существует равенство между энергией сигнала и энергией спектра.

2.2. Нейронная сеть

В качестве архитектуры нейронной сети рассмотрим предложенную Филипом Корзениовски в диссертации [1] сеть. Она показана на рисунке 2.3. Как можно увидеть, сеть является глубокой и состоит из трех скрытых полносвязных слоев из 512 узлов с функцией активации ReLU и выходного слоя из 12 узлов. В качестве входных данных используются пачки спектрограмм аудио-сигнала, получение которых было описано в пункте 2.1. Нейронная сеть не будет универсальной, поэтому ее нужно обучать для конкретных параметров, также описанных в пункте 2.1.

Функция активации *ReLU (Rectified Linear Unit)* чаще всего применяется при глубоком обучении и соответствует выражению

$$f(x) = \begin{cases} x, x > 0, \\ 0, x \leq 0. \end{cases} \quad (2.22)$$

Преимущества ReLU заключаются в следующем:

1) Производная подсчитывается очень легко и быстро: при $x < 0$ она равна

Название модели	Описание модели	Параметры модели для исследования
Модель $STFT$	Применение (2.7)/(2.17), возведенного в квадрат: $\left(\frac{STFT}{\sqrt{K//2+1}}\right)^2$, где K – размер окна.	• Размер окна, • Шаг окна, • Количество спектрограмм в пачке
Модель $MFCC$	Применение (2.9) + треугольные фильтры + логарифмирование + ДКП	• Размер окна, • Шаг окна, • Количество спектрограмм в пачке
Модель $fSTFT$	Модель STFT + треугольные фильтры	• Размер окна, • Шаг окна, • Количество спектрограмм в пачке, • Фильтры мел задаются обычные (40 штук)
Модель $mDCT$	Применение модифицированного ДКП с окном Vorbis (2.21)	• Размер окна, • Шаг окна, • Количество спектрограмм в пачке, • Фильтры 83 штуки по октавам

Таблица 2.2. Описание моделей предобработки звукового сигнала

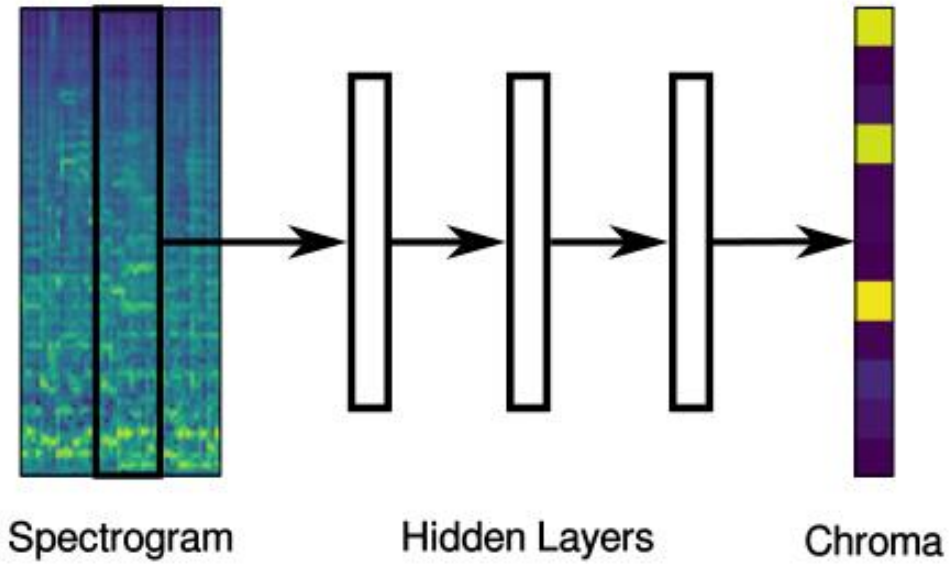


Рисунок 2.3. Модель сети Ф. Корзениовски

нулю, а при $x > 0$ – единице;

2) При ее использовании в силу характеристик функции количество включаемых в сеть нейронов уменьшается, за счет чего сеть становится легче.

По причине использования логистической регрессии как задачи классификации, которую я решаю в данном исследовании, выходной слой представляет собой softmax, который задается формулой (1.2). Количество узлов этого слоя соответствует 12 основным полутонам.

В качестве функции штрафа нейросети возьмем разреженную категориальную кросс-энтропию

$$loss = -\sum_{i=1}^{n} x_i \cdot \log\big(\sigma(y_i)\big), \tag{2.23}$$

где n – количество классов, x_i – целевые значения, y_i – спрогнозированные значения, $\sigma(y_i)$ – функция softmax для перевода в вероятности. Обучение нейронной сети включает в себя передачу данных вперед через модель и сравнение прогнозов с метками истинности. Это сравнение осуществляется с помощью функции потерь. Разреженная категориальная кросс-энтропия обычно используется в сценариях мультиклассовой классификации наряду с обычной категориальной кросс-энтропии. При этом она имеет ту же функцию потерь, что и категориальная кросс-энтропия. Единственная разница заключается в том, как нам необходимо представить ожидаемые результаты. В случае категориальной кросс-энтропии результирующие данные

являются векторами с 1 для результата и нулями - в остальных случаях. В нашем случае данные закодированы в целочисленном формате, поэтому мы используем разреженный вариант.

Одна из задач, которая стоит перед нами теперь – минимизировать выбранную функцию штрафа на обучающей выборке при нашей структуре сети. Эта минимизация осуществляется на основе градиентного спуска. Часто для преодоления не-глобальных минимумов функции ошибки при обучении сетей используются различные модификации градиентного спуска. Так, например, мы в качестве оптимизатора нейронной сети будем использовать метод *Adam* - один из вариантов адаптивного градиентного спуска, с помощью которого происходит более точная подстройка значения скорости обучения α отдельно для каждого из параметров сети $\boldsymbol{\theta} = \left(\mathbf{W}(l), \mathbf{b}(l)\right)_{l=2}^{L}$ ($\mathbf{W}(l)$ – матрица весов на l-ом слое, $\mathbf{b}(l)$ – смещение на l-ом слое, L – количество слоев), что позволяет учитывать их индивидуальное влияние на значение целевой ошибки [4]. В методе Adam для каждого параметра θ из $\boldsymbol{\theta}$ отслеживается история изменений частной производной целевой функции $g(n) = \partial E(n)/\partial\theta(\mathrm{n})$ ($n = 1, 2, \ldots$) путем накопления величины

$$G(n) = \gamma G(n-1) + (1-\gamma)g^2(n), \quad G(0) = 0, \quad 0 \le \gamma < 1. \tag{2.24}$$

Аналогичным образом также накапливаются взвешенные средние значения самих частных производных $g(n) = \partial E(n)/\partial\theta(n)$:

$$M(n) = \beta M(n-1) + (1-\beta)g(n), \quad n = 1, 2, \ldots; \quad M(0) = 0, \quad 0 \le \beta < 1. \tag{2.25}$$

Правило, по которому осуществляется каждая итерация градиентного спуска в методе Adam, выглядит так:

$$\theta(n+1) = \theta(n) - \frac{\alpha}{\sqrt{\hat{G}(n) + \varepsilon}}\widehat{M}(n), \tag{2.26}$$

где $\alpha > 0$ – параметр скорости обучения, а

$$\hat{G}(n) = \frac{G(n)}{1-\gamma}, \quad \widehat{M}(n) = \frac{M(n)}{1-\beta} \tag{2.27}$$

- скорректированные значения (2.24) и (2.25) соответственно.

Теперь выберем метрику оценивания результата классификации. Это необходимо для того, чтобы понять, насколько хорошо модель способна предсказывать результат.

Самая известная и при этом простая метрика – accuracy (точность). Ее нормализованная формула выглядит так:

$$accuracy = \frac{1}{n} \cdot \sum_{i=1}^{n} \left[\left(y_{test}^{i} == y_{true}^{i} \right) \right], \tag{2.28}$$

где n – количество меток, y_{test} – вектор полученных меток, y_{true} – вектор верных меток. В данной интерпретации значение $accuracy$ ранжируется от 0 до 1. Чем ближе его значение к 1, тем лучше работает модель. Задача такой метрики – показать долю правильных ответов. В исследовании будет использоваться именно эта метрика.

Для ускорения работы сети я использую пакетный режим подачи данных в сеть, то есть я буду объединять последовательные элементы набора данных в совокупности – пакеты – с определенным количеством элементов в каждом. В исследовании я рассматриваю длину пакета, равную 64. Пакетная обработка позволит ускорить процесс получения решения и точнее оценить вектор градиента.

ГЛАВА 3. РЕЗУЛЬТАТЫ ИССЛЕДОВАНИЯ

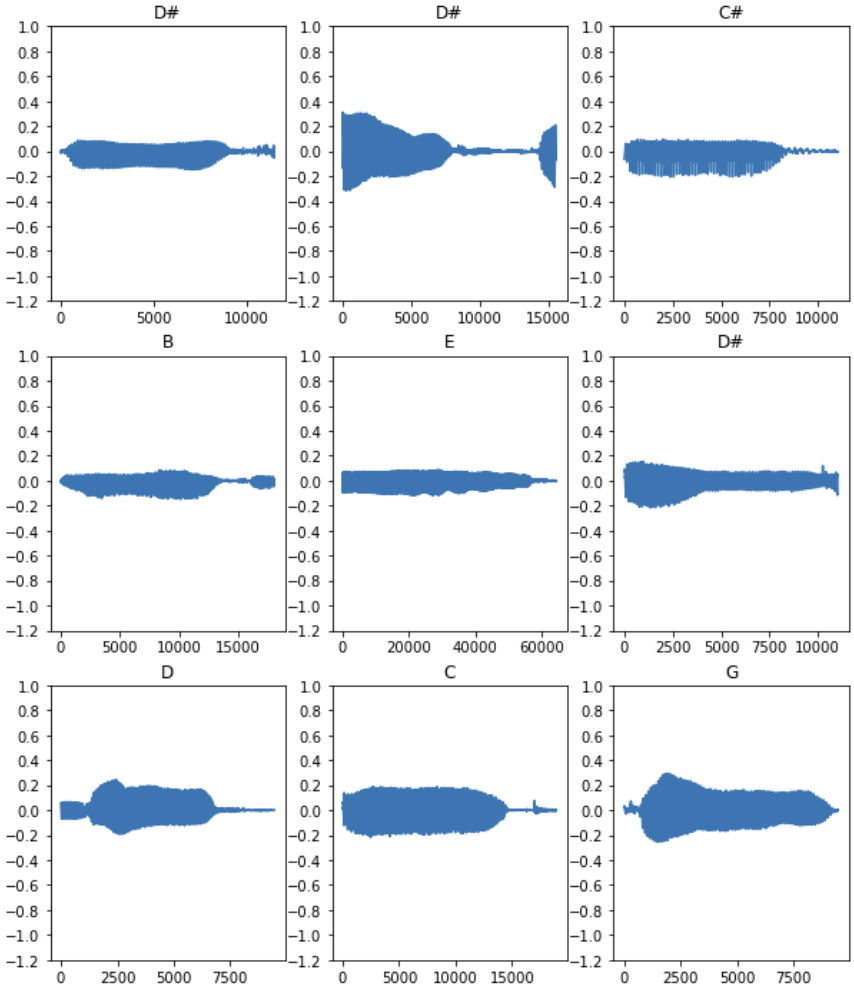


Рисунок 3.1. Примеры рассматриваемых звуковых сигналов, полученных с применением метода разбиения баз данных из п. 2.1.1

База данных, полученная согласно пункту 2.1.1 главы 2, состоит из 30828 звуковых моно-записей формата wav. На рисунке 3.1 показаны несколько примеров аудиосигналов – звучаний нот, с которыми можно было бы далее работать над получением спектрограмм. Тем не менее, не все эти данные будут учитываться. Согласно описанию выбора размера окна из пункта 2.1.3, необходимо стремиться к тому, чтобы окно было близко к размеру 11025. Однако, на практике я столкнулась с ситуацией, что часть файлов имеет размер, куда меньший, чем желаемое окно. В связи с этим мною было проведено небольшое исследование, задачей которого являлось выявление наиболее оптимального размера окна, при котором количество файлов, которыми можно пожертвовать и выбросить из рассмотрения из-за их короткой длины, было бы приемлемым. Для начала рассматривались размеры 2048, 4096, 8192, 10240 и

Размер окна	Процент выбрасываемых файлов	Количество выбрасываемых файлов
2048	0.036 %	11
4096	0.815 %	251
8192	11.348 %	3496
10240	19.957 %	6148
12600	41.544 %	12798

Таблица 3.1. Зависимость количества выбрасываемых файлом от размера окна

12600. Результаты показаны в таблице 3.1.

Из таблицы 3.1 видно неравномерное увеличение количества выбрасываемых файлов с увеличением размера окна. Теперь понятно, что размер окна, который изначально хотелось бы использовать (а именно 11025) не подходит, поскольку уже при размере 10240 выкидывается больше 6000 файлов, а это пятая часть всего набора данных. При малом объеме данных велика вероятность переобучения, поэтому таким количеством файлов жертвовать нельзя. Оптимальный процент находится в интервале, соответствующем промежутку между размерами окна 4096 и 8192. Значит, рассмотрим этот отрезок подробнее. Возьмем 100 точек – размеров окон, лежащих в данном промежутке, и посмотрим, как будет вести себя процент выбрасываемых файлов. Соответствующая зависимость продемонстрирована на рисунке 3.2. По графику видно квадратичный рост процента. В результате я остановилась на размере окна 8192. После

Рисунок 3.2. График зависимости процента выбрасываемых файлов от размера окна в промежутке от 4096 до 8192

выброса 3496 файлов, которые не удовлетворяют выбранному окну, в наборе осталось 27310 аудиозаписей.

Для обучения данные были разделены на 3 выборки: обучающую, проверочную и тестовую. Разбиение данных происходило в соотношении 70:20:10, поэтому для удобства рассматривалось только 27000 записей.

С учетом главы 2 у меня получилось 4 модели для рассмотрения. Архитектура каждой из них представлена на рисунках 3.3 – 3.6.

```
Model: "model_23"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 inputs (InputLayer)         [(None, 20, 4097)]        0

 flatten_23 (Flatten)        (None, 81940)             0

 relu_1 (Dense)              (None, 512)               41953792

 relu_2 (Dense)              (None, 512)               262656

 relu_3 (Dense)              (None, 512)               262656

 dense_end (Dense)           (None, 12)                6156

=================================================================
Total params: 42,485,260
Trainable params: 42,485,260
Non-trainable params: 0
_________________________________________________________________
```

Рисунок 3.3. Архитектура модели STFT

```
Model: "model_53"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 inputs (InputLayer)         [(None, 15, 40)]          0

 flatten_53 (Flatten)        (None, 600)               0

 relu_1 (Dense)              (None, 512)               307712

 relu_2 (Dense)              (None, 512)               262656

 relu_3 (Dense)              (None, 512)               262656

 dense_end (Dense)           (None, 12)                6156

=================================================================
Total params: 839,180
Trainable params: 839,180
Non-trainable params: 0
_________________________________________________________________
```

Рисунок 3.4. Архитектура модели MFCC

```
Model: "model_136"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 inputs (InputLayer)         [(None, 25, 83)]          0

 flatten_136 (Flatten)       (None, 2075)              0

 relu_1 (Dense)              (None, 512)               1062912

 relu_2 (Dense)              (None, 512)               262656

 relu_3 (Dense)              (None, 512)               262656

 dense_end (Dense)           (None, 12)                6156

=================================================================
Total params: 1,594,380
Trainable params: 1,594,380
Non-trainable params: 0
_________________________________________________________________
```

Рисунок 3.5. Архитектура модели fSTFT

```
Model: "model_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 inputs (InputLayer)         [(None, 3, 4096)]         0

 flatten_5 (Flatten)         (None, 12288)             0

 relu_1 (Dense)              (None, 512)               6291968

 relu_2 (Dense)              (None, 512)               262656

 relu_3 (Dense)              (None, 512)               262656

 dense_end (Dense)           (None, 12)                6156

=================================================================
Total params: 6,823,436
Trainable params: 6,823,436
Non-trainable params: 0
_________________________________________________________________
```

Рисунок 3.6. Архитектура модели mDCT

Для того, чтобы избежать переобучения, был использован механизм ранней остановки за счет отслеживания производительности во время обучения с помощью проверочного набора данных. Помимо этого была применена задержка на 10 эпох, чтобы предотвратить слишком раннюю остановку. Максимальное количество эпох – 300.

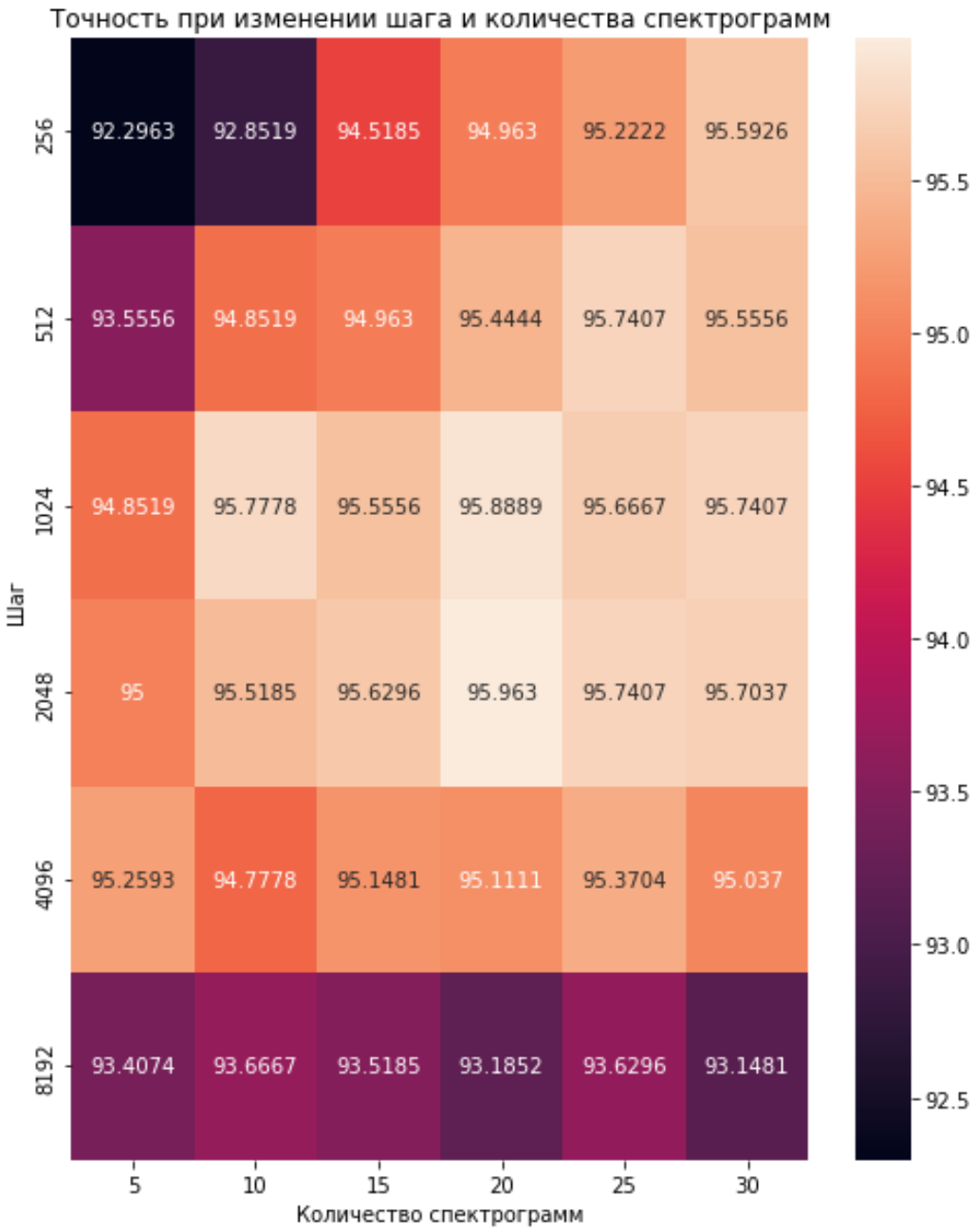


Рисунок 3.7. Карта точностей при изменении шага окна и количества спектрограмм на входе нейронной сети для модели STFT

Рассмотрим результаты применения модели STFT. Сначала осуществлялся подбор параметров предобработки. Как ранее было сказано, размер окна - фиксированный для всех моделей и равен 8192 отсчетам. Другими параметрами модели STFT являются шаг окна и количество спектрограмм в пачке, подаваемой на вход нейросети. Подбор выполнялся за счет сравнения точности модели при различных конфигурациях значений шага и количества спектрограмм. Так, шаг рассматривался равным 256, 512, 1024, 2048, 4096 и 8192, а количество спектрограмм – 5, 10, 15, 20, 25

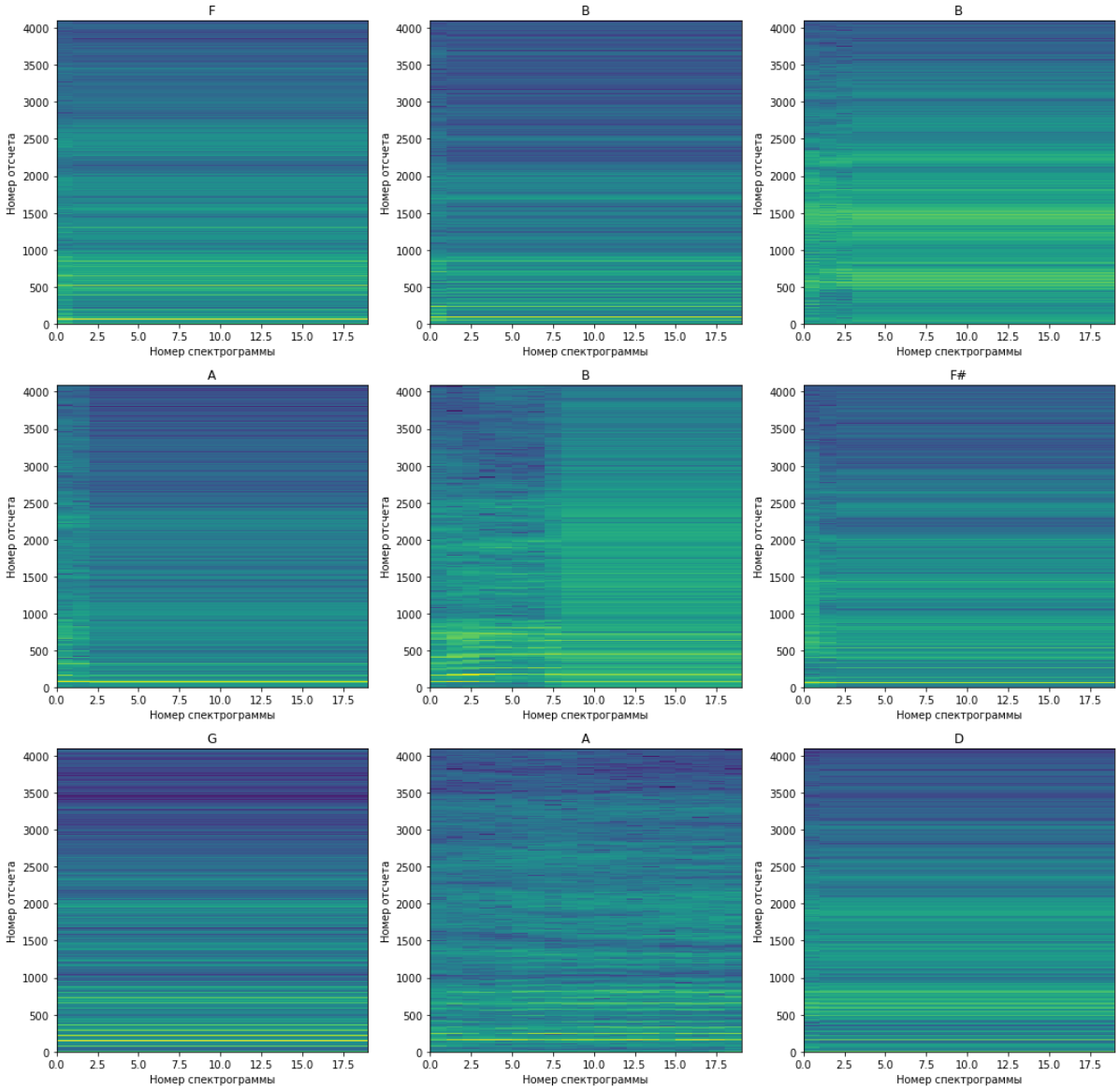


Рисунок 3.8. Примеры спектрограмм, полученные с применением подхода модели STFT (шаг окна – 2048, количество спектрограмм - 20)

и 30. Такие варианты параметров также применялись и в последующих моделях, за исключением модели mDCT. Рисунок 3.7 демонстрирует карту точностей для пар значений шага и количеств спектрограмм. Чем светлее клетка, тем выше точность. Из нее сразу становится понятно, что лучшими параметрами будут 2048 в качестве шага окна и 20 спектрограмм в пачке. Точность модели тогда будет равна 95.96 %. Время предобработки составило 5 секунд, а время обучения – 10 минут 55 секунд. На рисунке 3.8 показан пример спектрограмм, полученных для некоторых аудиозаписей из используемой базы данных. Согласно подходу модели STFT, применяемому для построения спектрограммы, по оси абсцисс откладывается номер спектрограммы, а по оси ординат – номер отсчета.

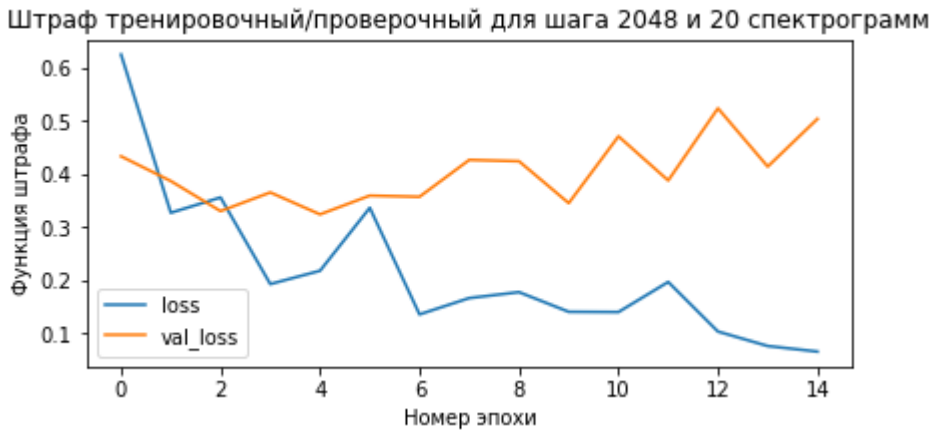


Рисунок 3.9. График сравнения штрафа на обучающей и проверочной выборках для модели STFT

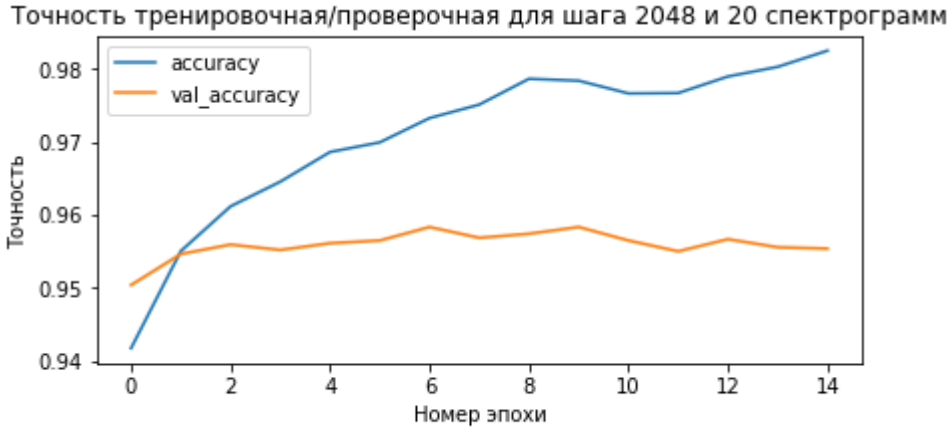


Рисунок 3.10. График сравнения точности на обучающей и проверочной выборках для модели STFT

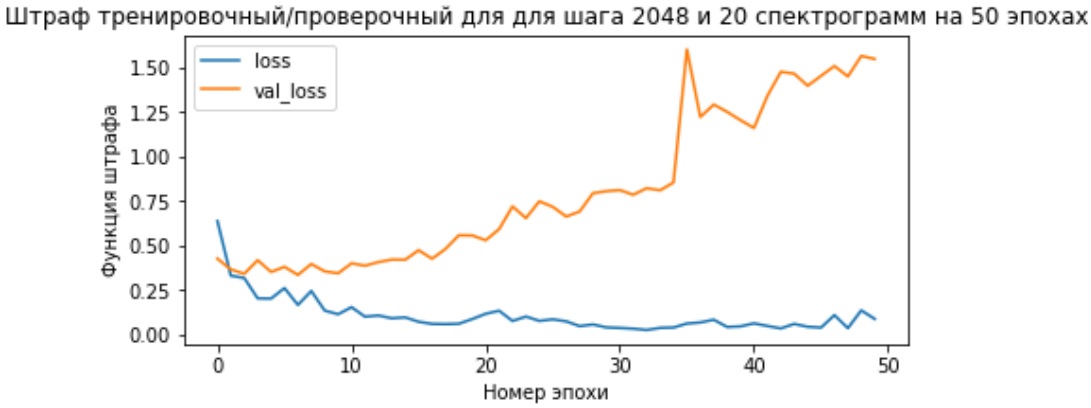


Рисунок 3.11. График сравнения штрафа на обучающей и проверочной выборках на 50 эпохах для модели STFT

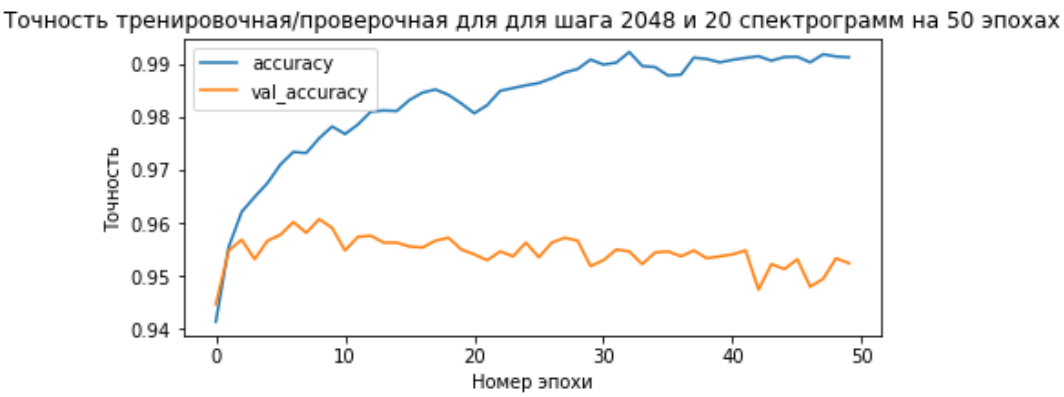


Рисунок 3.12. График сравнения точности на обучающей и проверочной выборках на 50 эпохах для модели STFT

Рисунок 3.13. Результат предсказания ноты E с помощью модели STFT

Рассмотрим штраф и точность обучения, которые показаны на рисунках 3.9 и 3.10 соответственно. Судя по всему, модель обучалась на 14 эпохах. При этом видно, что функция штрафа уже начинает подниматься, что говорит о начале переобучения.

Если рассматривать обучение на 50 эпохах, то по соответствующим графикам штрафа и точности (рисунки 3.11 и 3.12) можно заметить более явное переобучение. В то же время точность модели понижается до 95.11%.

Для проверки работоспособности сети была совершена попытка предсказания ноты E. Как показывает гистограмма на рисунке 3.13, эта проверка была успешно пройдена. Матрица ошибок, изображенная на рисунке 3.14 и показывающая наглядно результаты работы классификатора, также говорит о том, что в основном модель работает правильно: диагональ явно выражается по сравнению с остальными элементами.

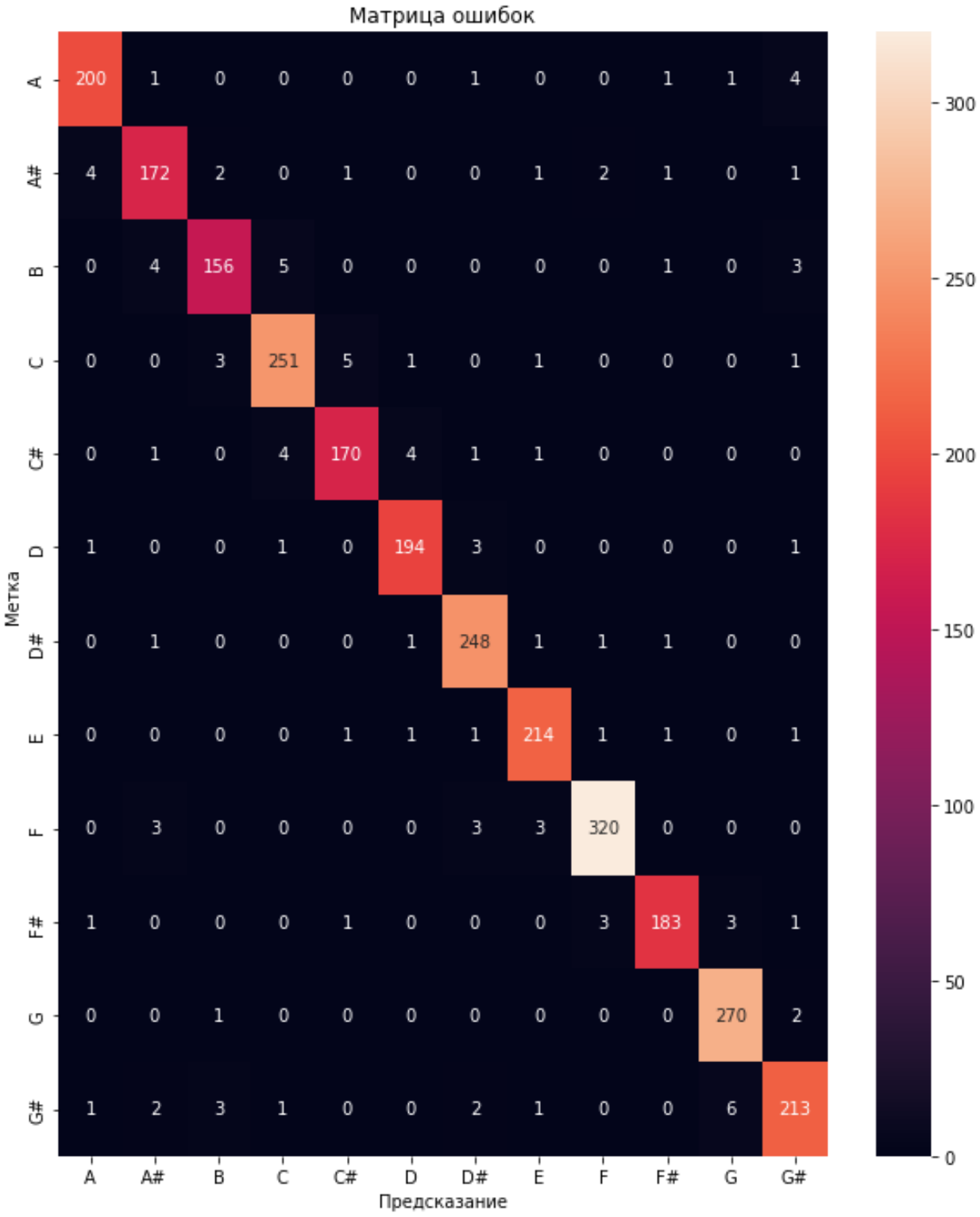


Рисунок 3.14. Матрица ошибок предсказаний модели STFT

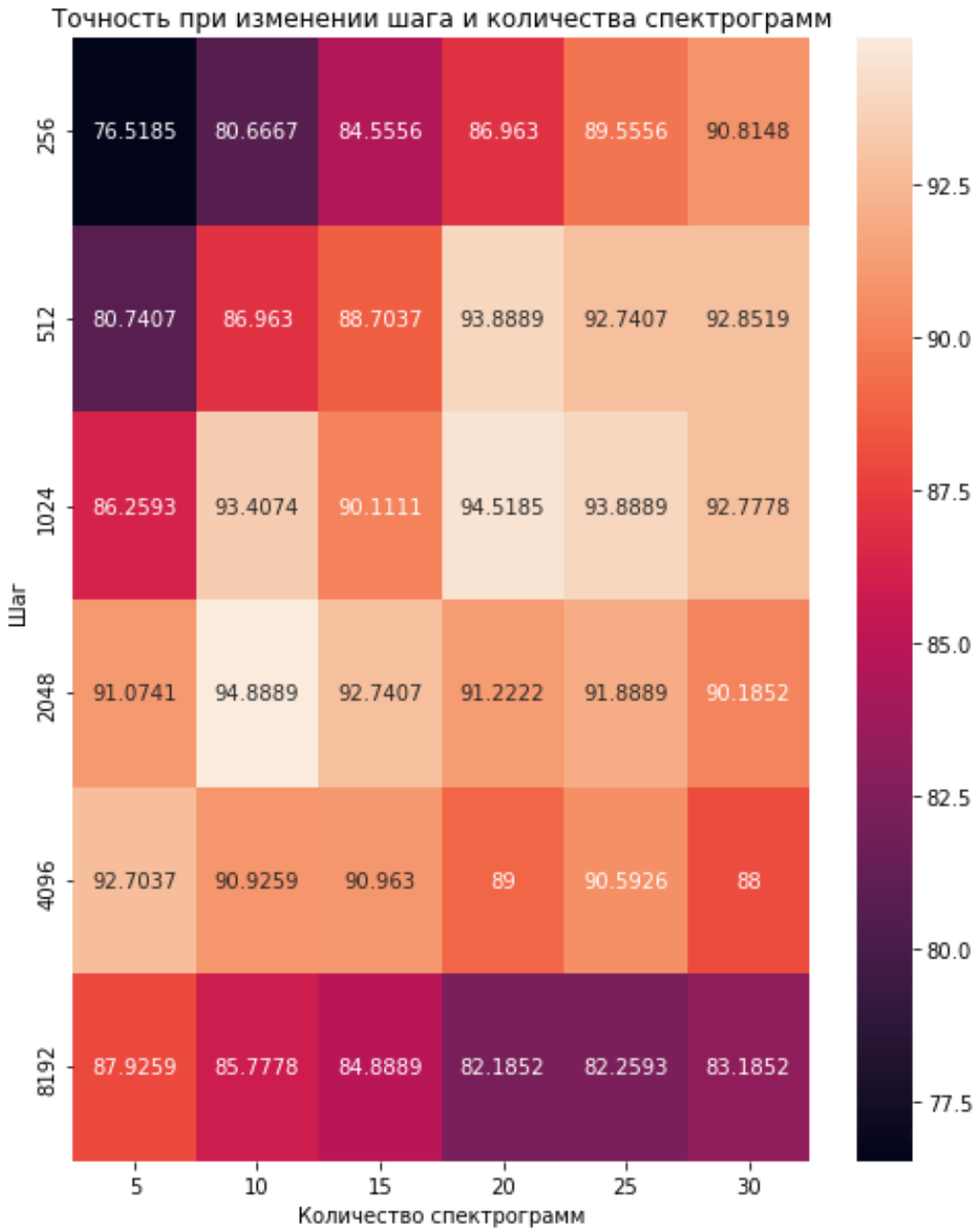


Рисунок 3.15. Карта точностей при изменении шага окна и количества спектрограмм на входе нейронной сети для модели MFCC

Рассмотрим результаты модели MFCC. Эта модель обычно используется именно для распознавания речи. В своем же исследовании я посмотрю, насколько она может быть хороша для распознавания нот. Подбор параметров предобработки совершался по подобию подбора для модели STFT. Результат можно увидеть на карте точностей, изображенной на рисунке 3.15. Судя по ней, наилучшая точность (94.89%) была достигнута при шаге окна 2048 и 10 спектрограммах в пачке. На рисунке 3.16 показаны примеры спектрограмм, полученные с применением MFCC при выбранных параметрах. Как можно увидеть, по оси абсцисс все еще отложены номера

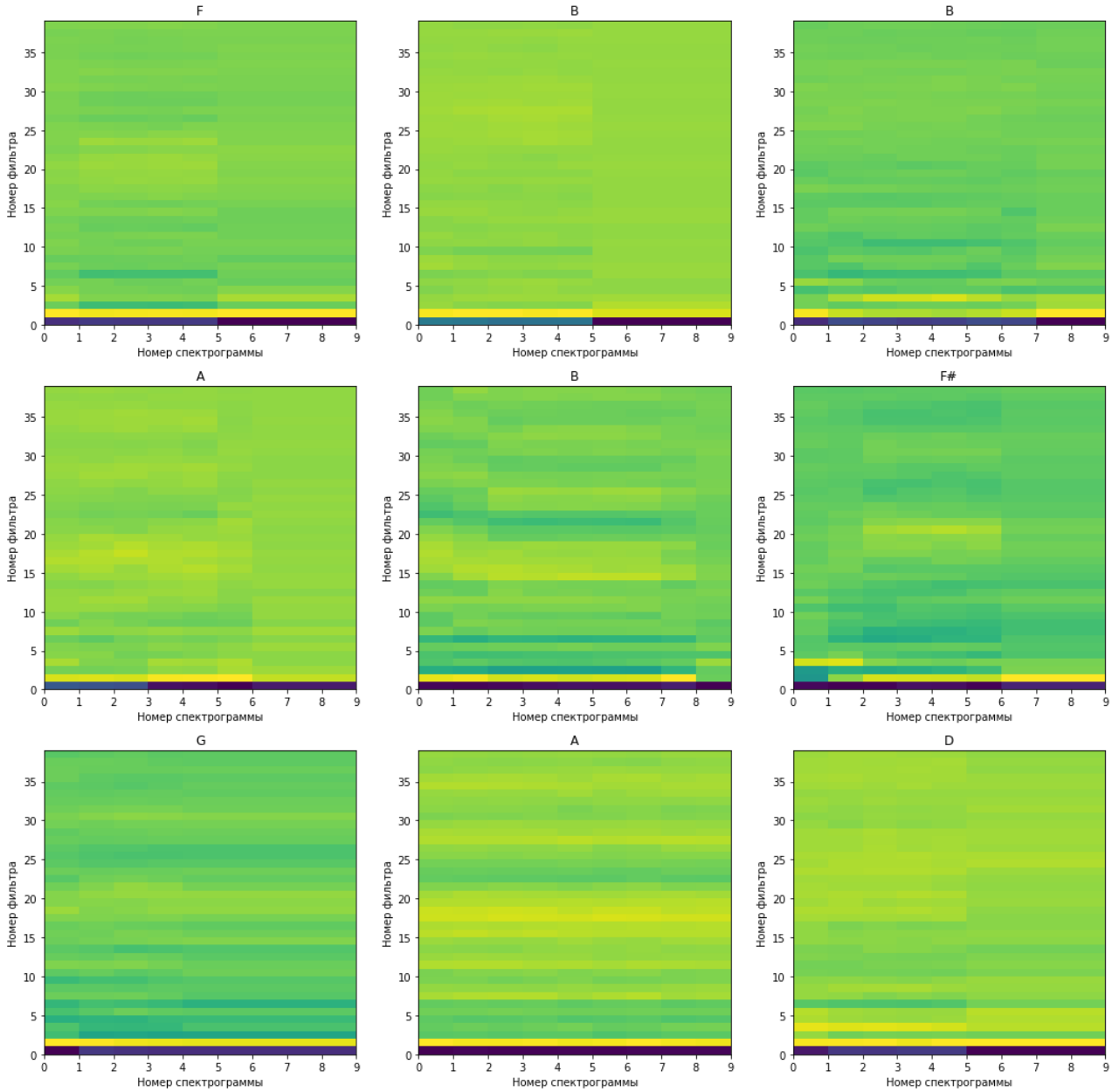


Рисунок 3.16. Примеры спектрограммы, полученные с применением подхода модели MFCC (шаг окна – 2048, количество спектрограмм - 10)

спектрограмм, но теперь ось ординат соответствует номерам треугольных фильтров, применяемых к мощности сигнала. К тому же становится очевидным влияние ДКП на результирующие спектрограммы: артефакты стали менее различимы по сравнению со спектрограммами модели STFT, что говорит о возможном ухудшении результатов распознавания.

Обучение совершалось на 17 эпохах. Время предобработки составило 3 секунды, а обучения – 1 минуту 18 секунд, что существенно быстрее модели STFT. Графики штрафа и точности при этом (рисунки 3.17 и 3.18) показывают себя более стабильными,

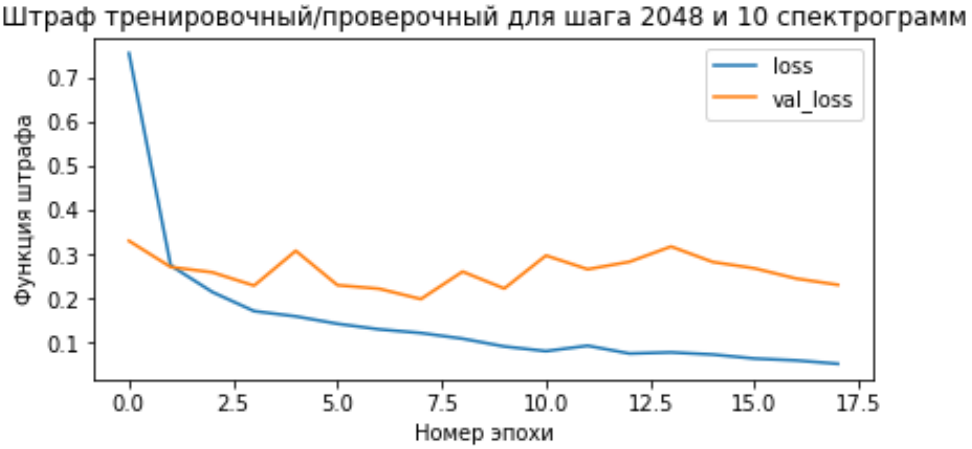


Рисунок 3.17. График сравнения штрафа на обучающей и проверочной выборках для модели MFCC

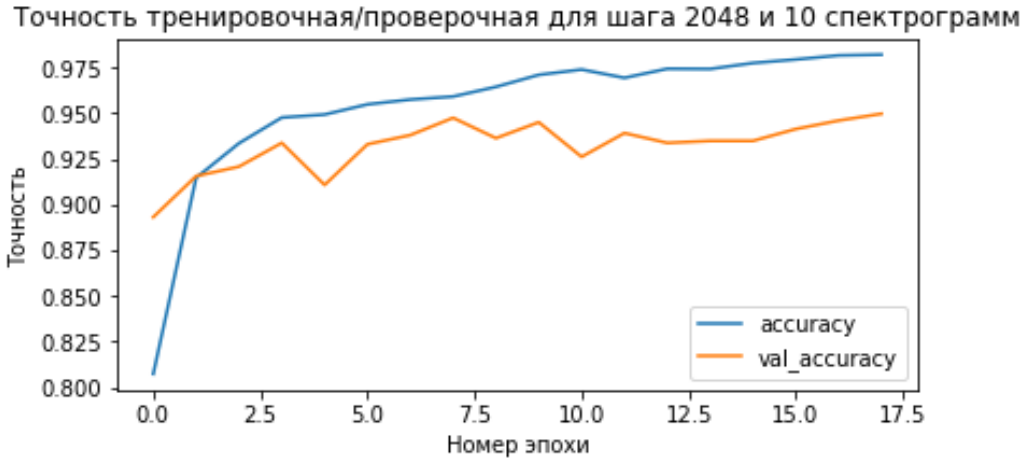


Рисунок 3.18. График сравнения точности на обучающей и проверочной выборках для модели MFCC

переобучение не наблюдается. Также рассмотрим обучение на 50 эпохах. Судя по штрафу и точности, показанным на рисунках 3.19 и 3.20, переобучение все же появляется, однако оно небольшое, график совсем немного начинает возрастать. Тем не менее, точность все же ухудшилась и теперь составляет всего лишь 93.70%.

Аналогично модели STFT была совершена проверка того, насколько текущая модель является рабочей. Гистограмма, показанная на рисунке 3.21, говорит нам о том, что проверка работоспособности на распознавании ноты E прошла успешно. По матрице ошибок для модели MFCC, которая продемонстрирована на рисунке 3.22, можно сказать, что эта модель неплохо определяет ноты, хотя и чаще ошибается на соседних нотах, чем модель STFT.

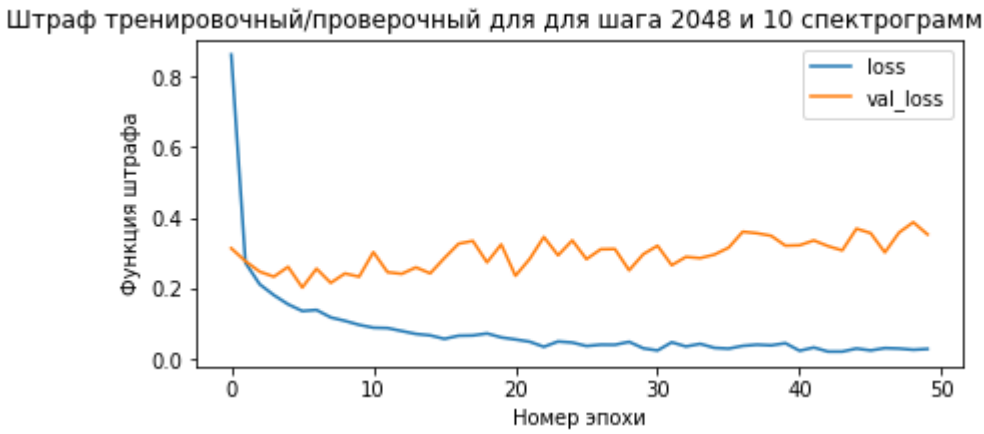


Рисунок 3.19. График сравнения штрафа на обучающей и проверочной выборках при обучении на 50 эпохах для модели MFCC

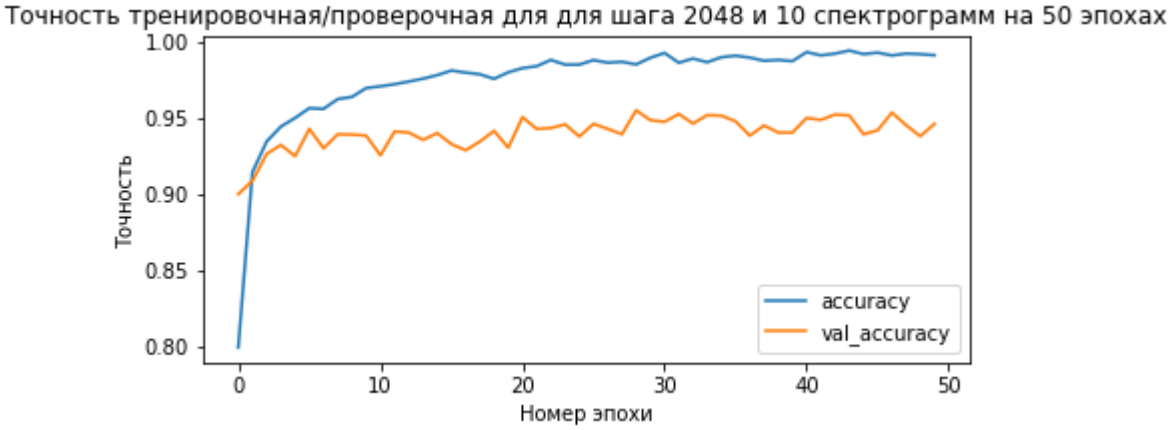


Рисунок 3.20. График сравнения точности на обучающей и проверочной выборках при обучении на 50 эпохах для модели MFCC

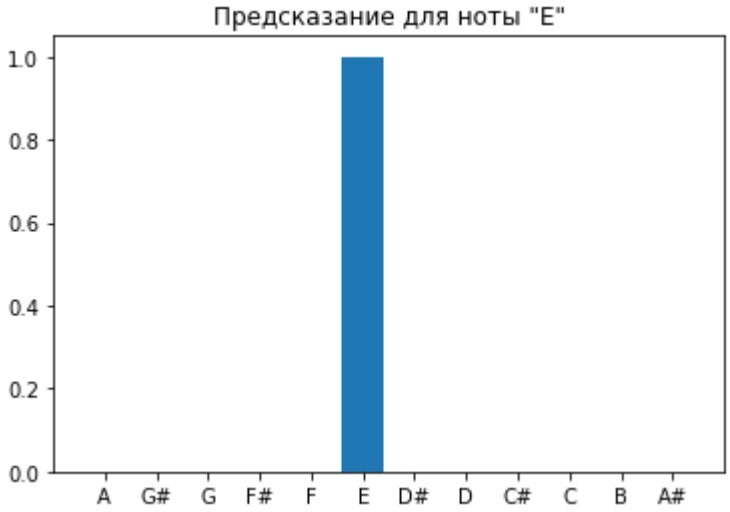


Рисунок 3.21. Результат предсказания ноты E с помощью модели MFCC

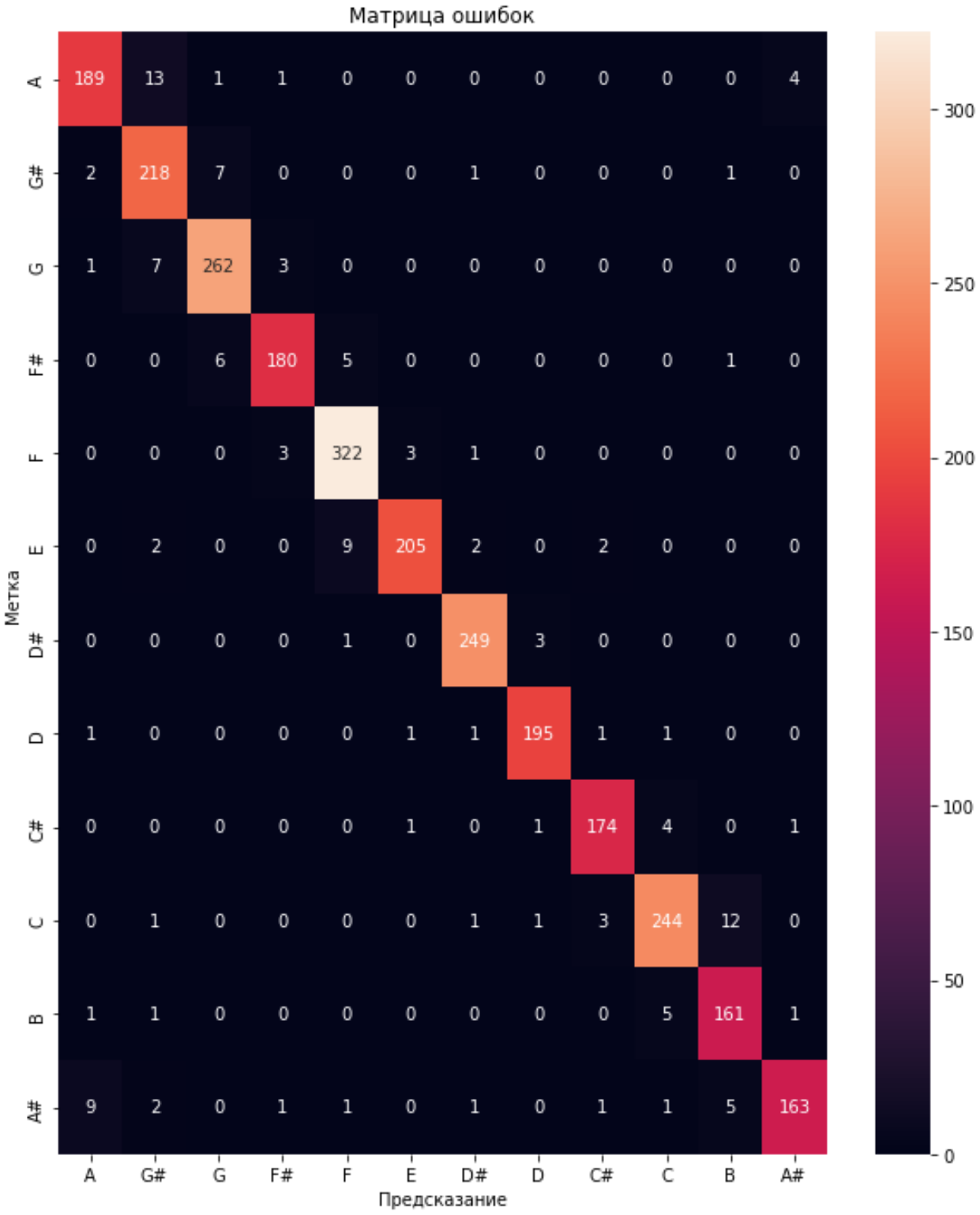


Рисунок 3.22. Матрица ошибок предсказаний модели MFCC

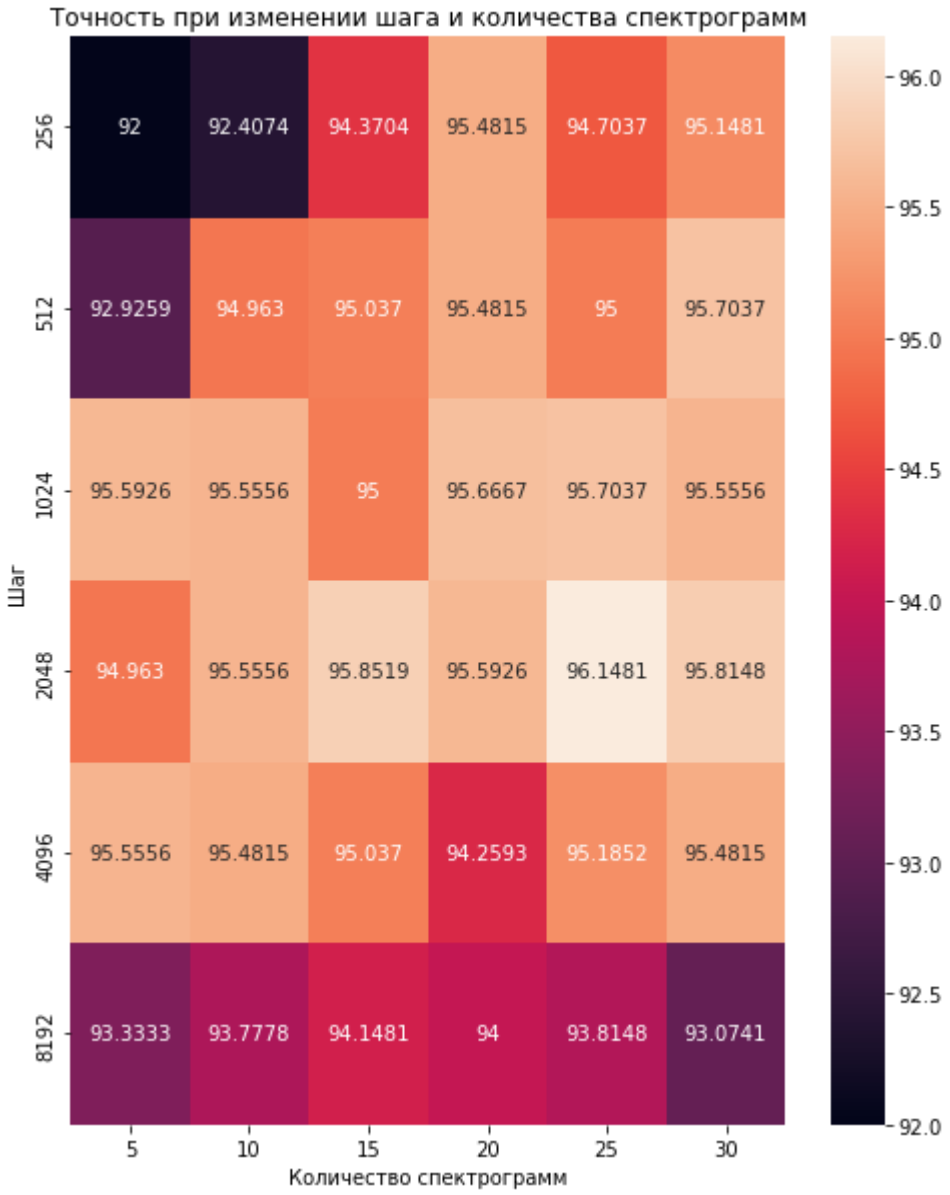


Рисунок 3.23. Карта точностей при изменении шага окна и количества спектрограмм на входе нейронной сети для модели fSTFT

Следующая модель, рассматриваемая в данном исследовании – fSTFT – вариация модели STFT, в которой применяется банк из 83 треугольных фильтров, получаемый вторым способом, описанным в пункте 2.1.3 главы 2. Поэтому получится сравнить, хорошо ли такой банк фильтров повлияет на точность модели или же нет.

Карта точностей, показанная на рисунке 3.23, говорит о том, что модель fSTFT превосходит модель STFT по точности. Так, лучший результат теперь равен 96.15%, что достигается при шаге окна 2048 и 25 спектрограммах. Рисунок 3.24 демонстрирует спектрограммы, полученные при таких параметрах предобработки. Артефакты на ней

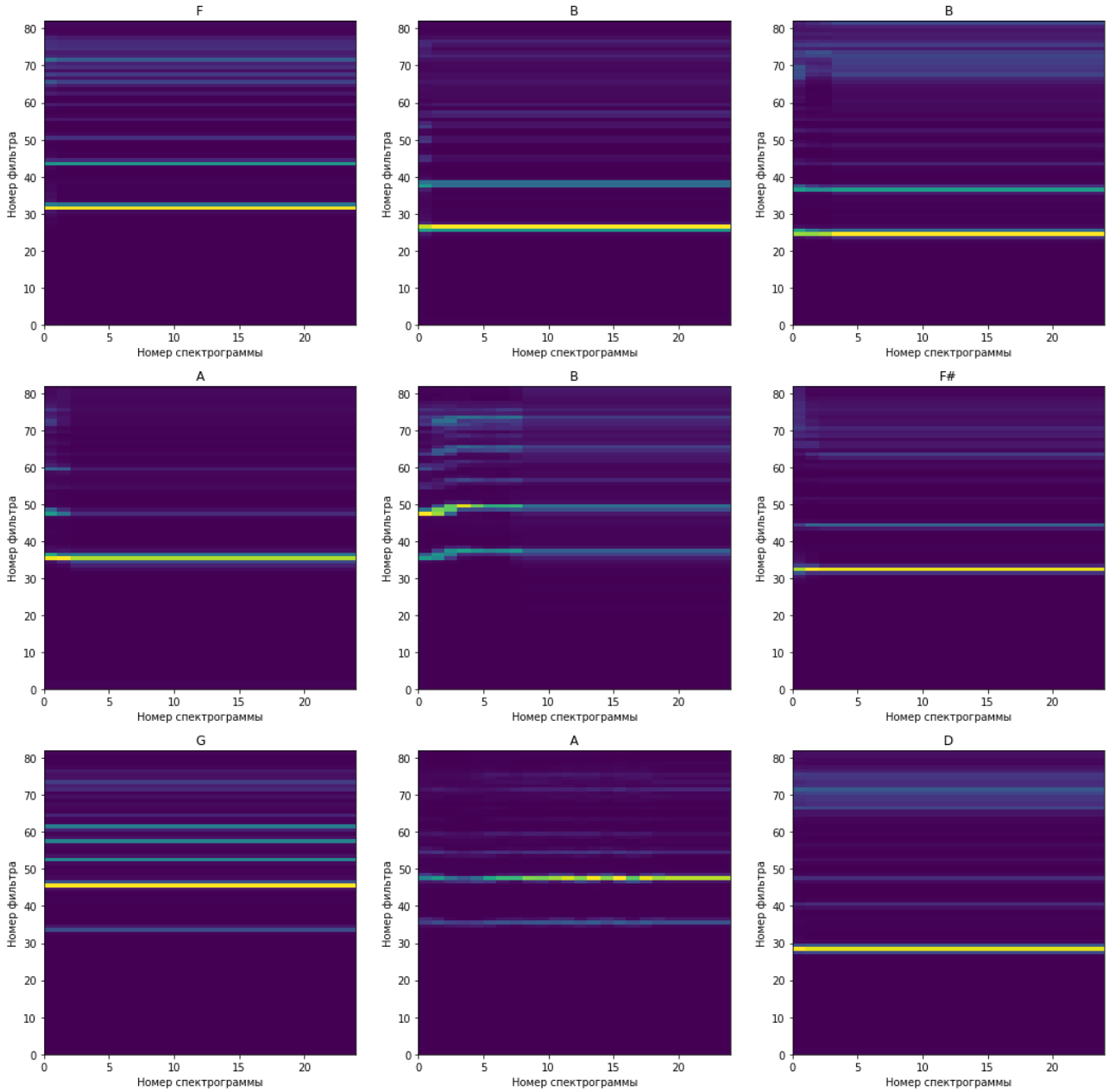


Рисунок 3.24 Примеры спектрограмм, полученные с применением подхода модели fSTFT (шаг окна – 2048, количество спектрограмм - 25)

теперь проявляются достаточно ярко. К тому же fSTFT – уже третья модель, для которой оптимальным является шаг окна, равный 2048.

Из-за ранней остановки количество эпох обучения – 13. Предварительная обработка заняла 2 секунды, обучения осуществилось 52 секунды. Модель fSTFT – самая быстрая из всех четырех моделей. На графиках штрафа и точности (рисунки 3.25 и 3.26) за 13 эпох очень явного переобучения не наблюдается, хотя штраф уже немного начинает расти. Но вот при обучении на 50 эпохах переобучение проявляется в полной

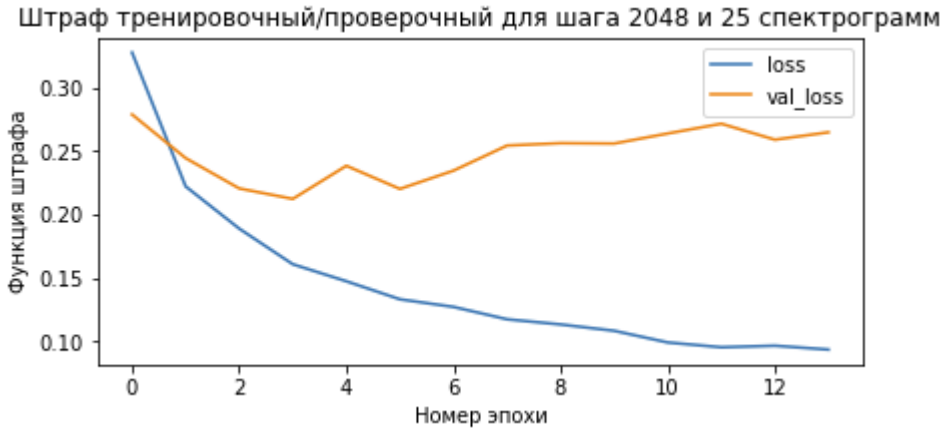


Рисунок 3.25. График сравнения штрафа на обучающей и проверочной выборках для модели fSTFT

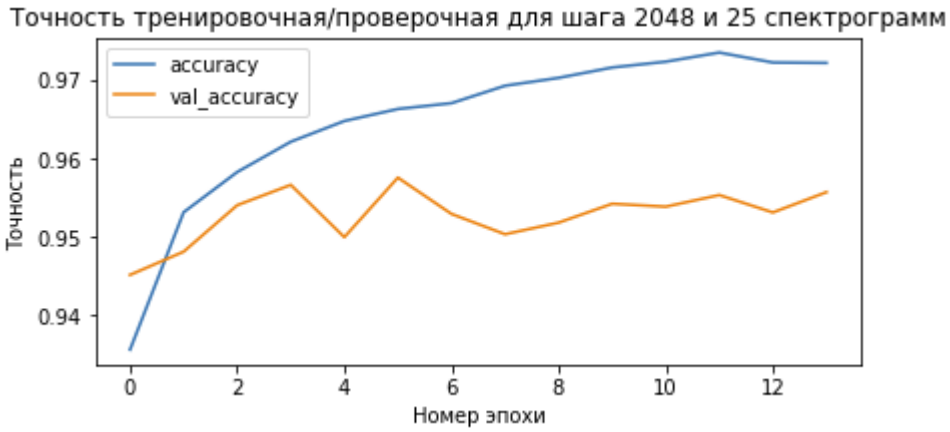


Рисунок 3.26. График сравнения точности на обучающей и проверочной выборках для модели fSTFT

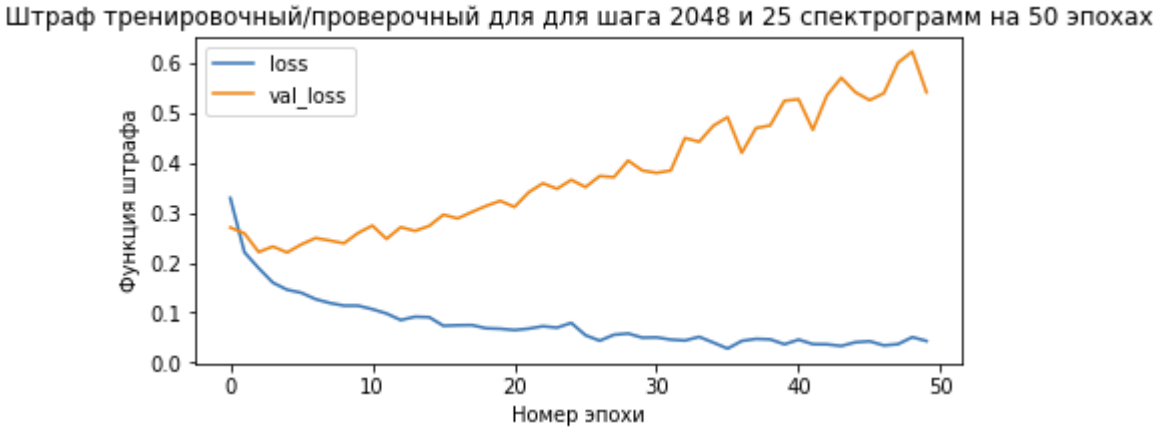


Рисунок 3.27. График сравнения штрафа на обучающей и проверочной выборках на эпохах 50 для модели fSTFT

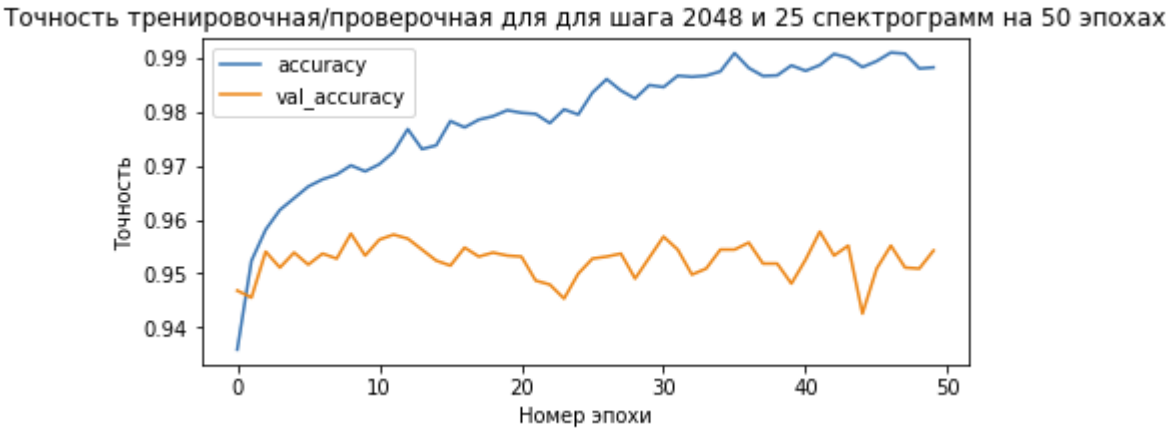


Рисунок 3.28. График сравнения точности на обучающей и проверочной выборках на 50 эпохах для модели fSTFT

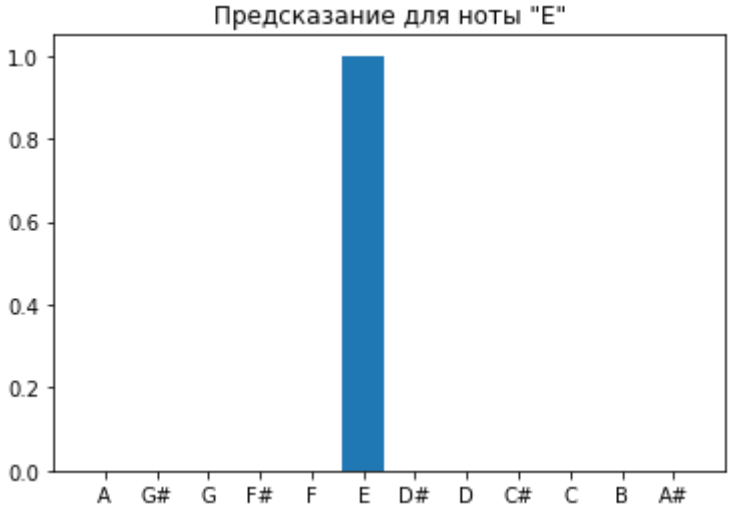


Рисунок 3.29. Результат предсказания ноты E с помощью модели fSTFT

мере, что демонстрируют рисунки 3.27 и 3.28. Точность при этом становится равной 95.30%. Проверка работоспособности проводилась и для этой модели. Так, нота E успешно была распознана, и это понятно по гистограмме на рисунке 3.29. На матрице ошибок (рисунок 3.30) четко прослеживается диагональ, поэтому можно сделать вывод, что сеть работает в значительном большинстве верно.

Последняя рассматриваемая модель – модель mDCT, которая использует вместо кратковременного преобразования Фурье модифицированное косинусное преобразование, описанное в пункте 2.1.4 главы 2.

Модель отличается от остальных тем, что шаг окна в ней предопределен – он равен половине размера окна. Поскольку для всех моделей окно имеет размер 8192 отсчета, то это значит, что шаг равен 4096. Перебираемым параметром остается

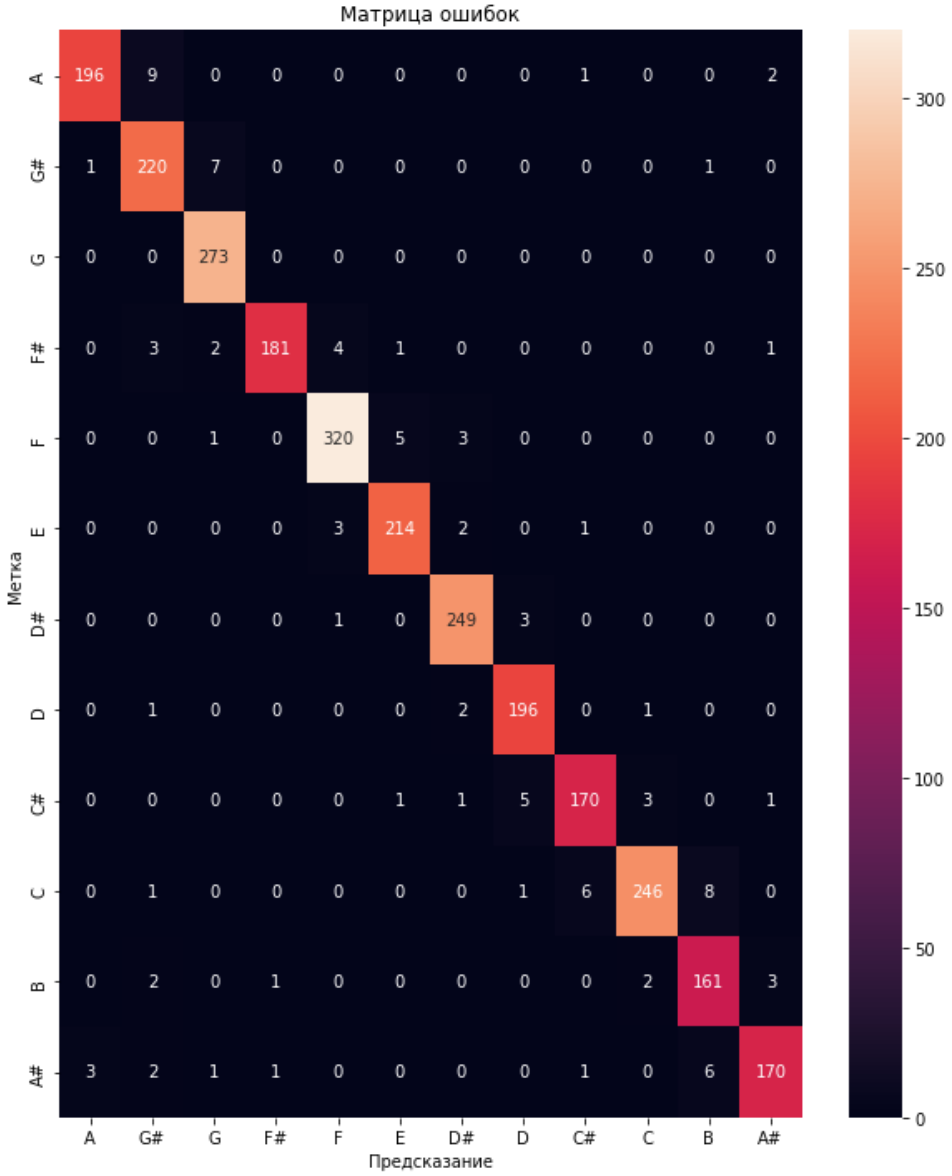


Рисунок 3.30. Матрица ошибок предсказаний модели fSTFT

Рисунок 3.31. Выбор количества спектрограмм в пачке на входе нейронной сети для модели mDCT

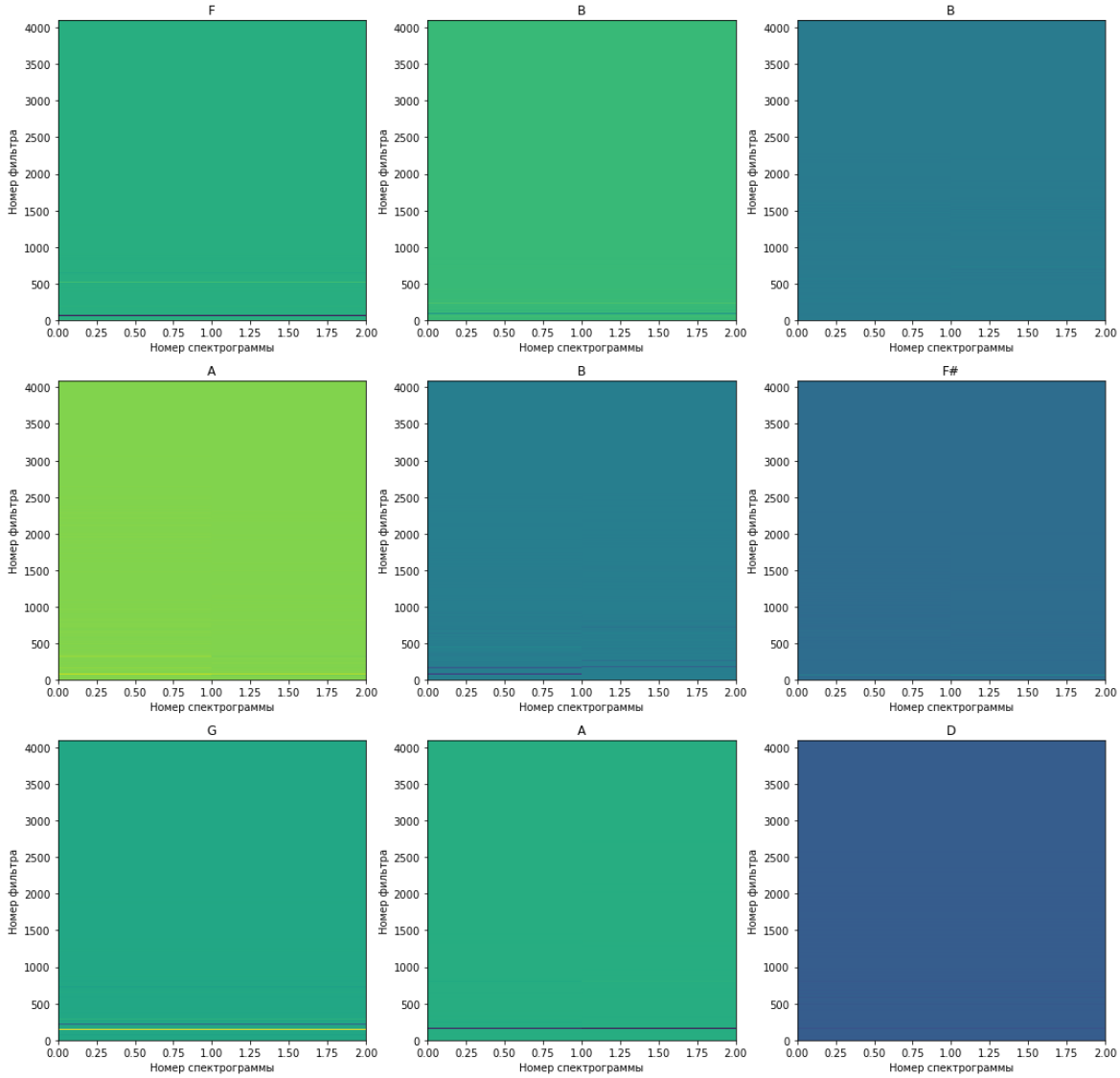


Рисунок 3.32. Примеры спектрограмм, полученные с применением подхода модели mDCT (количество спектрограмм - 3)

количество спектрограмм в пачке. Мною было рассмотрено 6 вариантов: 1, 3, 5, 10, 15, 20. Выбираем то количество спектрограмм, которое дает наилучшую точность. Судя по зависимости, отображенной на рисунке 3.31, необходимо выбрать 3 спектрограммы, поскольку с увеличением их количества точность начинает падать. Точность при этом будет равна 88.55%, предобработка проходит за 2 секунды, а обучение – за 1 минуту 20 секунд.

Спектрограммы, построенные при полученном параметре для модели mDCT, приводятся на рисунке 3.32. По нему сразу становится понятно, что результаты текущей модели не будут лучшими, поскольку артефактов для распознавания практически нет.

Следом за этим рассмотрим штраф и точность. Рисунки 3.33 и 3.34 дают понять, что обучение с учетом ранней остановки проводилось на 10 эпохах. Поскольку задержка остановки составляла 10 эпох, можно сделать вывод, что переобучение начинается практически сразу. Эту же мысль подтверждают графики штрафа и точности на 50 эпохах (рисунки 3.35 и 3.36). Точность так же падает до 87.78%.

По стандарту модель тестировалась на распознавание ноты E, и этот тест был успешно пройден (рисунок 3.37). Также была построена матрица ошибок, которая говорит о небольшом увеличении числа ошибочных предсказаний соседних нот по сравнению с остальными моделями (рисунок 3.38).

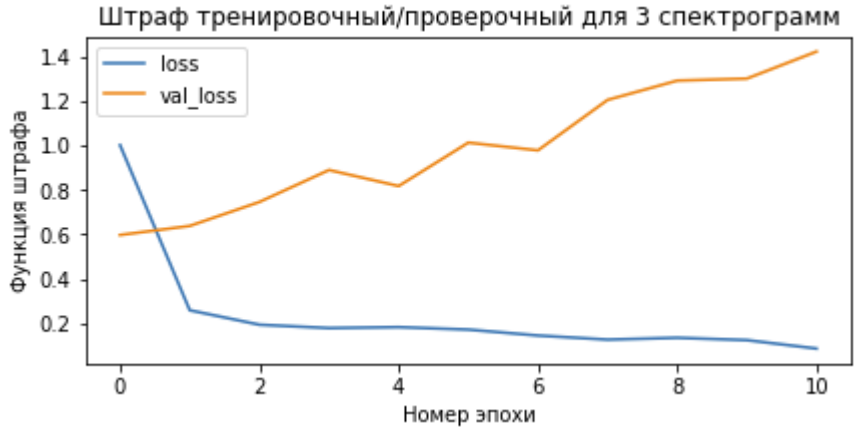


Рисунок 3.33 График сравнения штрафа на обучающей и проверочной выборках для модели mDCT

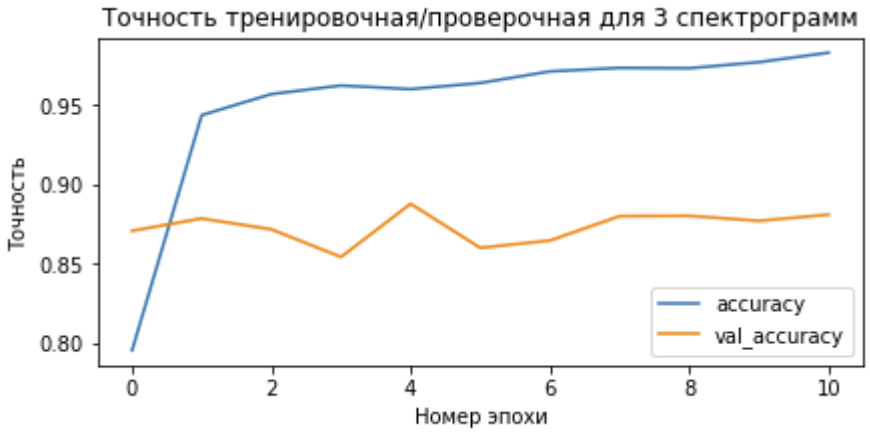


Рисунок 3.34. График сравнения точности на обучающей и проверочной выборках для модели mDCT

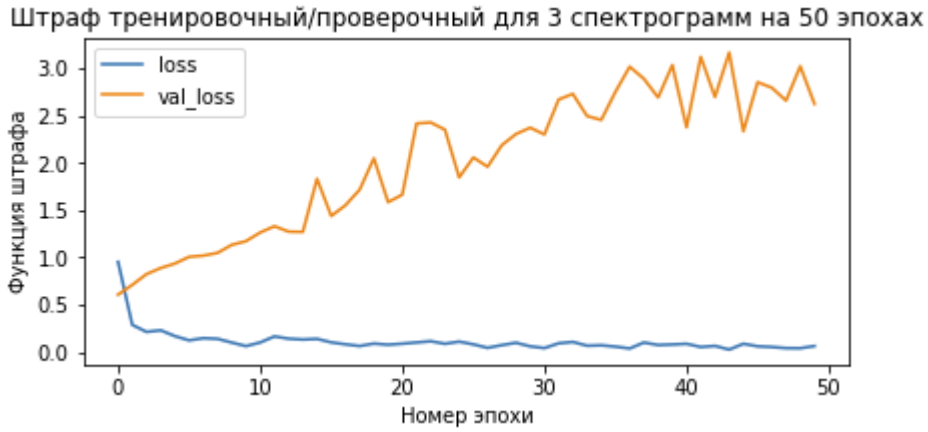


Рисунок 3.35. График сравнения штрафа на обучающей и проверочной выборках на эпохах 50 для модели mDCT

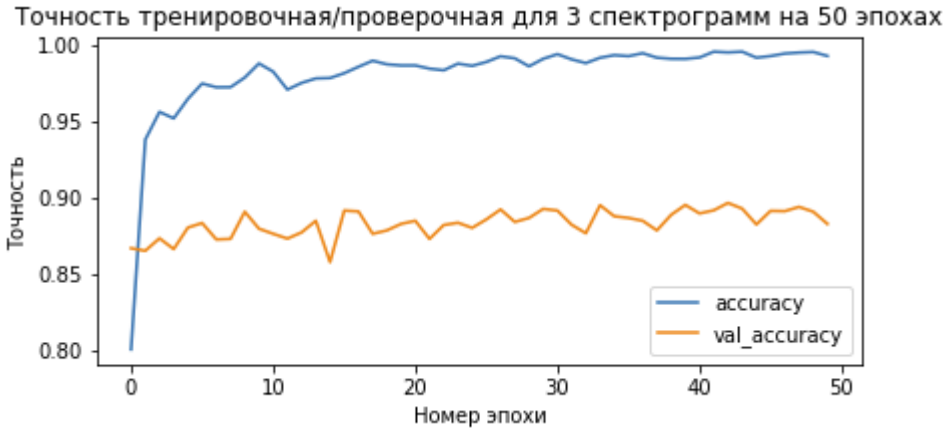


Рисунок 3.36. График сравнения точности на обучающей и проверочной выборках на 50 эпохах для модели mDCT

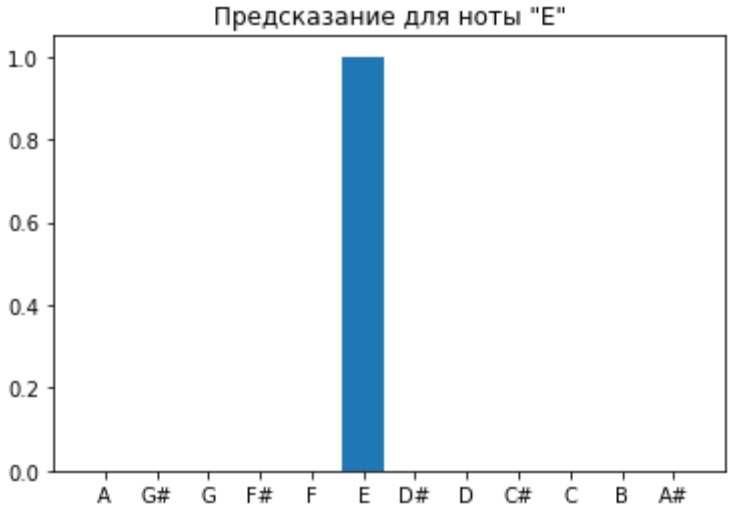


Рисунок 3.37. Результат предсказания ноты E с помощью модели mDCT

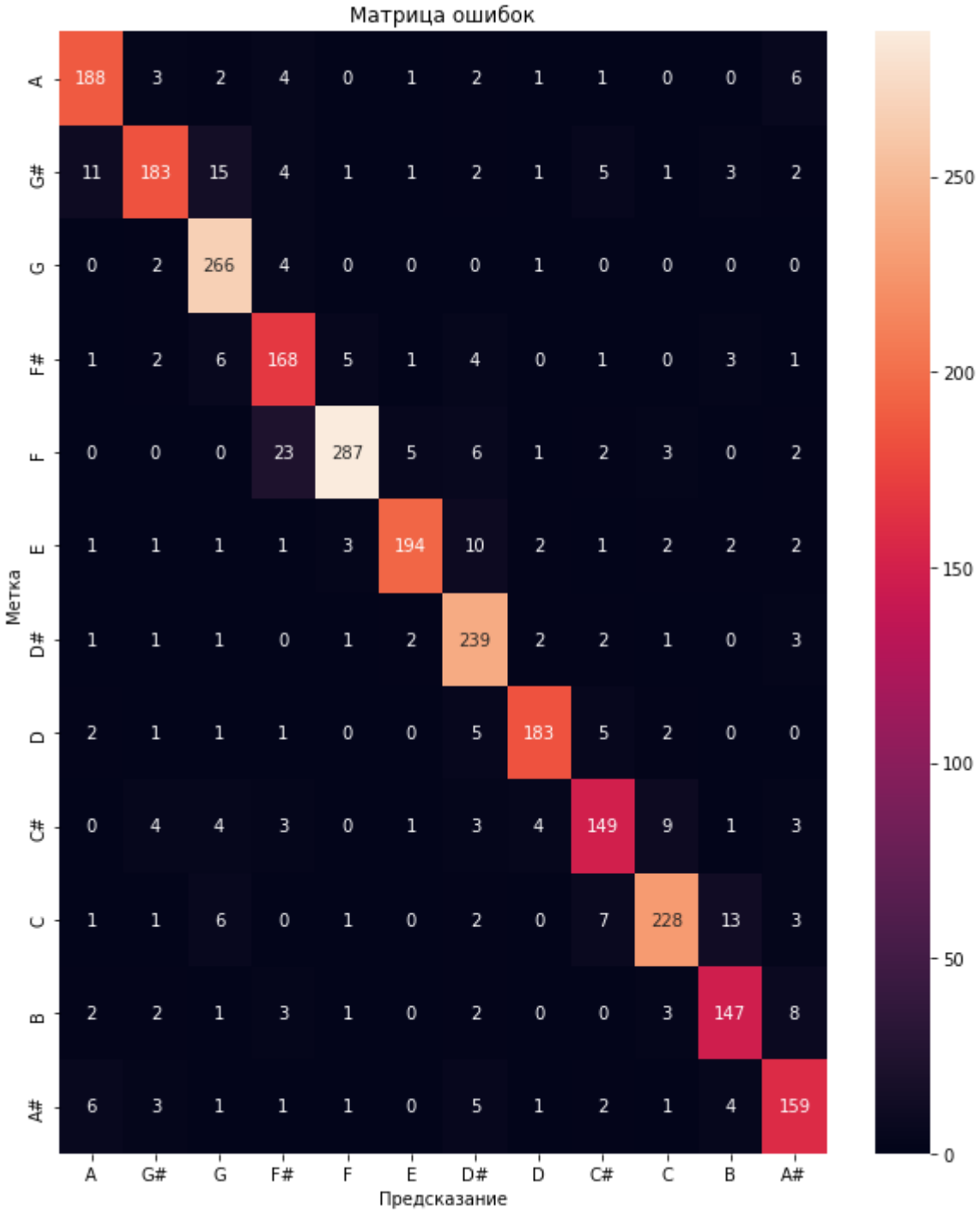


Рисунок 3.38. Матрица ошибок предсказаний модели mDCT

Таким образом, было исследовано 4 модели распознавания нот. Результаты исследований приведены в таблице 3.2. В совокупности можно сделать вывод, что наилучшей моделью является модель fSTFT как самая быстрая и точная.

Модель	Итоговая точность	Время предобработки	Время обучения	Параметры			Склонность к переобучению
				Размер окна	Шаг окна	Количество спектрограмм в пачке	
STFT	95.96%	5 сек	10 мин 55 сек	8192 отсчета	2048 отсчетов	20	Высокая
MFCC	94.04%	3 сек	1 мин 18 сек	8192 отсчета	2048 отсчетов	15	Низкая
fSTFT	96.15%	2 сек	52 сек	8192 отсчета	2048 отсчетов	25	Средняя
mDCT	88.55%	2 сек	1 мин 20 сек	8192 отсчета	4096 отсчетов (по умолчанию)	3	Средняя

Таблица 3.2. Результаты исследования по всем моделям

ЗАКЛЮЧЕНИЕ

В данной ВКР была рассмотрена задача разработки метода распознавания нот мелодии путем извлечения основных гармоник.

Первый этап исследования заключался в том, чтобы вывести, каким образом задача распознавания нот связана с задачей классификации в теории машинного обучения. На втором этапе отдельно рассматривалось 2 шага решения задачи распознавания нот: предобработка и построение нейронной сети. Также было выбрано несколько моделей, которые могут помочь решить поставленную проблему. На последнем этапе проводились практические исследования этих моделей, а также была выбрана лучшая из них.

Все практические работы проводились на базе данных нот, построенной в соответствии с методикой, описанной на втором этапе.

Однако, настоящее исследование может быть продолжено. Так, например, есть возможность рассмотрения других параметров предобработки, отличных от используемых в этой работе. Также можно развить теорию распознавания нот во времени и расширить само распознавание с опорой на различие октав.

СПИСОК ЛИТЕРАТУРЫ

1. F. Korzeniowski. Harmonic Analysis of Musical Audio using Deep Neural Networks // Linz, September 2018

2. E. A. Cantos, J. M. I. Quereda, J. J. V. Mas. Flute audio labelled database for Automatic Music Transcription // Zenodo, September 4, 2018

3. S. Choi, W. Kim, S. Park, S. Yong, J. Nam. CSD: Children's Song Dataset for Singing Voice Research (1.0) // International Society for Music Information Retrieval (ISMIR), Zenodo, May 25, 2021

4. С. В. Умняшкин. Основы теории распознавания образов и машинного обучения // НИУ МИЭТ, 2021, Москва

5. С. В. Умняшкин. Основы теории цифровой обработки сигналов // НИУ МИЭТ, 2016, Москва, с. 394 – 409

6. Марюфич М.Р. Автоматическое распознавание аккордов с использованием скрытых марковских моделей // 2015, Санкт-Петербург, с. 11 – 12

7. Алдошина И. А. Основы психоакустики // 2000, Санкт-Петербург, с. 5 – 7

8. S. Davis, P. Mermeltein. Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences // IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. 28 No. 4, pp. 357 – 366

9. X. Huang, A. Acero, H. Hon. Spoken Language Processing: A guide to theory, algorithm, and system development // Prentice Hall, 2001

10. Декарт, Р. Компендиум музыки. Музыкальная эстетика Западной Европы XVII-XVIII веков / Р. Декарт; пер. с лат. Л.А. Фрейденберга; сост. и общая вступительная статья В.П. Шестакова // М.: Музыка, 1971, с. 349

11. Y. Wang, AES Member, M. Vilermo. Modified Discrete Cosine Transform – Its Implications for Audio Coding and Error Concealment // J. Audio Eng. Soc., Vol. 51, No. 1/2, 2003 January/February, pp. 52 – 53

12. Rizwan M. LeNet-5-A Classic CNN Architecture // Data Science Central, 2018

13. W. Wang, Y. Li, T. Zou, X. Wang, J. You, Y. Luo. A Novel Image Classification Approach via Dense-MobileNet Models // Hindawi, Mobile Information System, January 6, 2020

14. Pujara A. Image Classification With MobileNet // Analytics Vidhya, July 4, 2020

15. Kurama V. A Review of Popular Deep Learning Architectures: AlexNet, VGG16, and GoogleNet // PaperspaceBlog, 2020