Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Институт кибернетики
Направление 230100 «Информатика и вычислительная техника»
Кафедра вычислительной техники

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Тема работы
РАСПОЗНАВАНИЕ МУЗЫКАЛЬНЫХ ИНСТРУМЕНТОВ НА ОСНОВЕ НЕЙРОСЕТЕВОГО АНАЛИЗА АМПЛИТУДНО-ЧАСТОТНЫХ ХАРАКТЕРИСТИК

УДК 004.93

Студент

Группа	ФИО	Подпись	Дата
8ВМ2А	Ф.В. Станкевич		

Руководитель

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент каф. ВТ	А.А. Белоусов	к.т.н.		

Томск – 2014 г.

РЕФЕРАТ

Магистерская диссертация содержит 98 страниц формата А4, 29 рисунков, 25 таблиц и 31 литературный источник.

Ключевые слова: музыкальные инструменты, классификация, искусственные нейронные сети, мел-кепстральные коэффициенты.

Объектом исследования является звук музыкальных инструментов.

Цель работы — разработка алгоритма распознавания звука музыкальных инструментов с высокой точностью распознавания и производительностью при минимизации числа используемых признаков.

В работе предлагается способ решения задачи, основанный на нейросетевом распознавании признаков, извлеченных из амплитудно-частотного спектра звукового сигнала — мел-кепстральных коэффициентов (mel frequency cepstral coefficients). Данные признаки представляют собой набор вещественных чисел, характеризующих форму звукового спектра, а именно распределение энергии сигнала вдоль его спектра. Для снижения сложности пространства признаков предлагается использовать метод главных компонент. Распознавание признаков производиться с помощью искусственной нейронной сети прямого распространения. Архитектура сети определяется с помощью нейроэволюционного алгоритма.

В результате применения предложенного метода удалось решить задачу распознавания музыкальных инструментов и получить высокую точность при малом числе признаков.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность работы. Задача распознавания музыкальных инструментов востребована при обработке сложных музыкальных сигналов, а именно аудиозаписей музыкальных произведений различных жанров, таких как классическая музыка, эстрадная музыка и другие. Задача распознавания музыкальных инструментов относиться к классу задач Music Information Retrieval (MIR), и может быть востребована в следующих областях:

- Аннотирование медиаконтента (метаданные)
    - для генерации плейлистов
    - для использования в поисковых системах
- Сегментация музыкальных сигналов
- Идентификация музыкальных объектов

Также данная задача может рассматриваться как первый этап решения задачи сегментации сложных музыкальных сигналов и идентификации музыкальных объектов (нот).

Цель работы. Целью работы является разработка алгоритма распознавания звука музыкальных инструментов с высокой точностью распознавания и производительностью при минимизации числа используемых признаков.

Предмет исследования. Предметом исследования является звук музыкальных инструментов, его структура и свойства. Особое внимание уделяется амплитудно-частотным характеристикам звука. В данной работе рассматривается как звук отдельно звучащих музыкальных инструментов, так и звук музыкальных инструментов звучащих на фоне других музыкальных звуков в сложных сигналах (то есть сольные партии музыкальных инструментов).

Научная и практическая новизна. Разработан и реализован алгоритм, позволяющий распознавать звуки различных музыкальных инструментов с высокой точностью и производительностью при минимальной размерности пространства признаков.

Практическая значимость результатов. Разработанный алгоритм может быть использован для аннотирования медиаконтента и как составной компонент решения задачи сегментации сложных музыкальных сигналов и идентификации музыкальных объектов (нот).

Апробация работы. Результаты работы были опубликованы в трудах X Международной научно-практической конференции «Ключевые аспекты научной деятельности–2014» (Польша, 2014 г.), XX Международной научно-практической конференции «Современные техника и технологии» (г. Томск, 2013 г.), I Международной научно-практической конференции «Информационные технологии в науке, управлении, социальной сфере и медицине» (г. Томск, 2013 г.).

1 МЕТОДЫ И ОБЗОР ЛИТЕРАТУРЫ

Человеческий мозг способен распознавать различные визуальные и звуковые образы, в том числе и отличать звучание одного музыкального инструмента от другого, даже на фоне звучания других инструментов. В данной работе рассматривается проблема распознавания тембра отдельного музыкального инструмента на основе его спектральных особенностей с применением нейронных сетей.

Звук представляет собой волну, распространяющуюся в среде и изменяющуюся во времени. Любая волна характеризуется спектром и амплитудой. Каждый музыкальный инструмент имеет свое неповторимое звучание, которое характеризуется набором обертонов содержащихся в звуке данного музыкального инструмента. Схематическое изображение основного тона и обертонов во временно́м диапазоне представлено на рисунке 1.1.

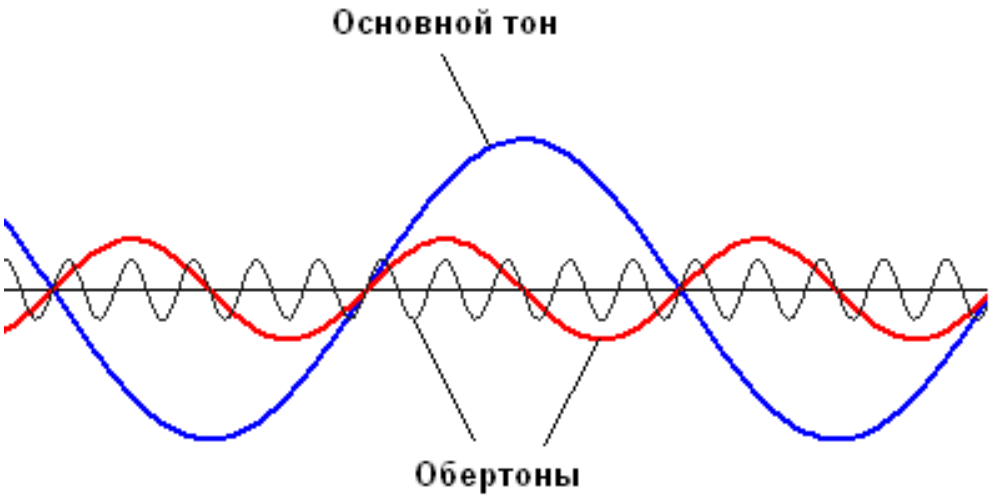

Рисунок 1.1 Основной тон и обертоны [1]

Подробное описание музыкального звука, его свойств и обзор литературы выполнены на английском языке и представлены в приложении А.

2 РАСПОЗНАВАНИЕ МУЗЫКАЛЬНЫХ ИНСТРУМЕНТОВ НА ОСНОВЕ НЕЙРОСЕТЕВОГО АНАЛИЗА АМПЛИТУДНО-ЧАСТОТНЫХ ХАРАКТЕРИСТИК

2.1. Мел-частотные кепстральные коэффициенты

Главной задачей в статистическом распознавании является выделение и извлечение признаков. Под выделением признаков понимается процесс, в котором пространство данных преобразуется в пространство признаков.

Основной характеристикой отличающей звук одного музыкального инструмента от другого являются обертоны (дополнительные частоты, содержащиеся в звуке) и их изменение во времени. Информацию об обертонах целесообразно извлекать из спектральной характеристики звукового сигнала с учетом амплитуды спектральных составляющих и использовать такую информацию для решения задачи распознавания. Однако спектр сигнала представляет собой большой набор данных, а именно все частоты, содержащиеся в сигнале, что является избыточным для поставленной зачади.

Вследствие избыточности информации в спектре сигнала, видится целесообразным, снизить размерность информации и выделить признаки из пространства данных. Для этой цели предлагается использовать мел-частотные кепстральные коэффициенты (mel-frequency cepstral coefficients, MFCC) – набор коэффициентов, характеризующих сигнал на основе его спектра и амплитуды. Данные коэффициенты успешно применяются в задачах распознавания речи, а также зарекомендовали себя как признаки, дающие наибольшую точность при распознавание инструментов [2, 3, 4 и 5].

Мел-частотные кепстральные коэффициенты являются представлением сигнала, определяемым на основе его спектральной плотности. Спектральная плотность сигнала рассчитывается с применением оконной функции, размер окна нелинейно изменяется. Данный подход аппроксимирует поведение слуховой системы человека. Впервые мел-частотные кепстральные коэффициенты были предложены Дэвисом и Мермельштейном [6], которые

показали эффективность их использования для решения задач распознавания речи.

Алгоритм вычисления мел-частотных кеспральных коэффициентов представляется следующими этапами:

1. Разбиение входного сигнала на короткие промежутки (окна).
2. Вычисление спектра сигнала для каждого окна на основе преобразования Фурье.
3. Расчет спектральной плотности с применением мел-шкалы.
4. Вычисление логарифма спектральной плотности.
5. Выполнение дискретного косинусного преобразования (ДКП).
6. Отбрасывание второй половины полученных коэффициентов.

Теперь рассмотрим некоторые этапы приведенного алгоритма более подробно.

Разбиение входного сигнала. Аудио сигнал постоянно изменяется во времени, для упрощения работы предположим, что сигнал постоянен на некотором малом временном интервале (постоянен в статистическом смысле, т.е. статистически стационарный, т.к. очевидно, что даже на малом временном интервале сигнал всегда изменяется). Поэтому мы разбиваем входной сигнал на промежутки (окна) размером 20-100 мс.

Расчет спектральной плотности сигнала. В человеческом ухе находится орган — улитка, которая вибрирует в различных частях в зависимости от частоты звука. Эти зоны вибрации воздействуют на разные нервные окончания, позволяя мозгу анализировать и интерпретировать звуковой сигнал.

Полный спектр сигнала содержит большое количество избыточной информации, не требуемой для решения задач распознавания сигнала. В частности, улитка не может воспринимать разницу между близко расположенными частотами. Этот эффект становиться более значимым при возрастании частоты. По этой причине весь частотный разбивается на участки, затем вычисляется суммарная энергия участка, причем участки частично

перекрываются. Размеры этих участков рассчитываются при помощи мел-шкалы: первое окно очень узкое и показывает, сколько энергии содержится около нуля герц. По мере возрастания частоты размер окна увеличивается, так как изменения в высокочастотном диапазоне менее различимы человеческим ухом. Более подробное описание данного этапа будет приведено ниже.

Вычисление логарифма. После вычисления энергии, мы вычисляем ее логарифм. Это также моделирует слуховую систему человека: мы воспринимаем громкость в нелинейной шкале — для удвоения воспринимаемой громкости необходимо увеличить энергию в 8 раз.

Выполнение дискретного косинусного преобразования. Последним шагом является выполнения дискретного косинусного преобразования для полученных логарифмов энергий. Есть две основные причины для выполнения данного этапа. Так как участки частот, для которых вычисляется энергия, частично перекрываются, то они достаточно сильно коррелируют друг с другом. ДКП устраняет данную корреляцию. Также высшие коэффициенты ДКП представляют резкие изменения энергии. Эти резкие изменения ухудшают точность распознавания звуковых образов, поэтому старшие коэффициенты ДКП отбрасываются, что позволяет повысить производительность распознавания.

Мел-шкала соотносит воспринимаемую частоту, или высоту, чистого тона к реальной частоте звука. Люди хорошо замечают небольшие изменения частоты в низкочастотном диапазоне, однако в высокочастотном диапазоне изменения частоты для человека менее заметны. Таким образом, мел-шкала адаптирует частоты к человеческому восприятию.

Преобразования частоты в мел-шкалу выполняется согласно формуле (2.1).

$$M(f) = 1125 \cdot \ln(1 + \frac{f}{700}) \quad (2.1)$$

Обратное преобразование выполняется по формуле (2.2).

$$M^{-1}(m) = 700 \cdot (e^{\frac{m}{1125}} - 1) \quad (2.2)$$

На рисунке 2.1 представлена зависимость мел-частоты от фактической частоты.

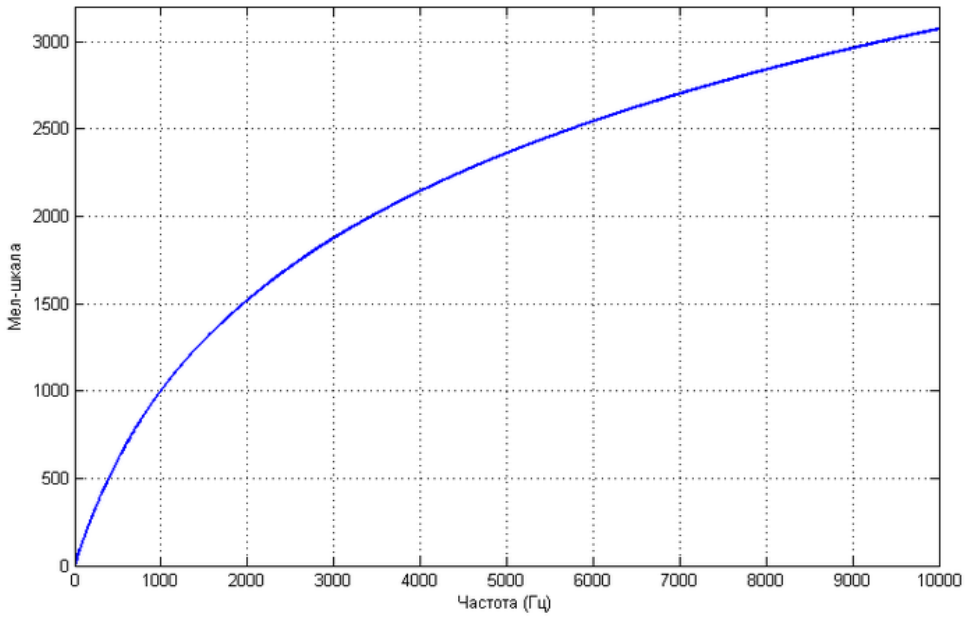

Рисунок 2.1. Зависимость мел-частоты от фактической частоты звука

Рассмотрим расчет коэффициентов более подробно. В расчетах будем предполагать, что наш сигнал имеет частоту 16КГц.

В первую очередь необходимо разбить сигнал на отрезки (frames, фреймы), например по 25 мс каждый. Каждый такой фрейм содержит 0.025*16000 = 400 отсчетов (samples, сэмплов). Шаг между двумя фреймами обычно меньше, чем размер фрейма, например 10 мс (160 сэмплов), что позволяет достичь перекрытия соседних фреймов. Это значит, что если первый фрейм начинает с нулевого сэмпла, то второй фрейм будет начинаться со 160го сэмпла, при длине фрейма в 400 сэмплов.

Следующий шаг это расчет 12 коэффициентов MFCC для каждого фрэйма. Обозначим исходный сигнал через $s(n)$, тогда каждый фрейм есть $s_i(n)$, где n есть длина сигнала и изменяется от 1 до 400 (при размере фрейма в 400 сэмплов), а i есть количество фреймов.

Первым этапом для расчета MFCC является выполнение преобразования Фурье. Пусть $S_i(k)$ есть результат преобразования для i-го фрейма, вычисляемая

по формуле (2.3), а $P_i(k)$ — спектральная плотность, вычисляемая по формуле (2.4).

$$S_i(k) = \sum_{n=1}^{N} s_i(n)h(n)e^{-\frac{j\pi kn}{N}}, 1 \leq k \leq K \quad (2.3)$$

где K есть длина преобразования,

$h(n)$ — оконная функция (например, функция Хэмминга),

N — размер оконной функции.

$$P_i(k) = \frac{1}{N}|S_i(k)|^2 \quad (2.4)$$

Результат (2.4) называется *периодограмма*, она оценивает спектральную плотность сигнала. Для ее расчета мы берем абсолютное значение комплексного числа, полученного в результате преобразования Фурье, а также отбрасываем вторую половину коэффициентов Фурье из-за их симметрии.

Следующим этапом при расчете MFCC является использование мел-шкалы для расчета коэффициентов. Для этих целей используются так называемые треугольные фильтры, которые мы применяем к периодограмме, полученной на предыдущем этапе. Количество таких фильтров может варьироваться от 20 до 40, обычно используется 26 фильтров.

Фильтр представляет собой вектор, большинство значений которого нули, лишь на определенном участке вектор имеет значения отличные от нуля. Для применения фильтра мы перемножаем данный вектор на периодограмму. Данный фильтр показывает, сколько энергии приходиться на определенный участок спектра. Иллюстрация использования фильтров представлена на рисунке 2.2: *a* — полный набор фильтров (заметим, что фильтры расположены с перекрыванием), *b* — пример энергетического спектра сигнала, *c* — восьмой фильтр, *d* — энергетический сектор в восьмом фильтре, *e* — 20-й фильтр, *f* — энергетический спектр в 20-ом фильтре.

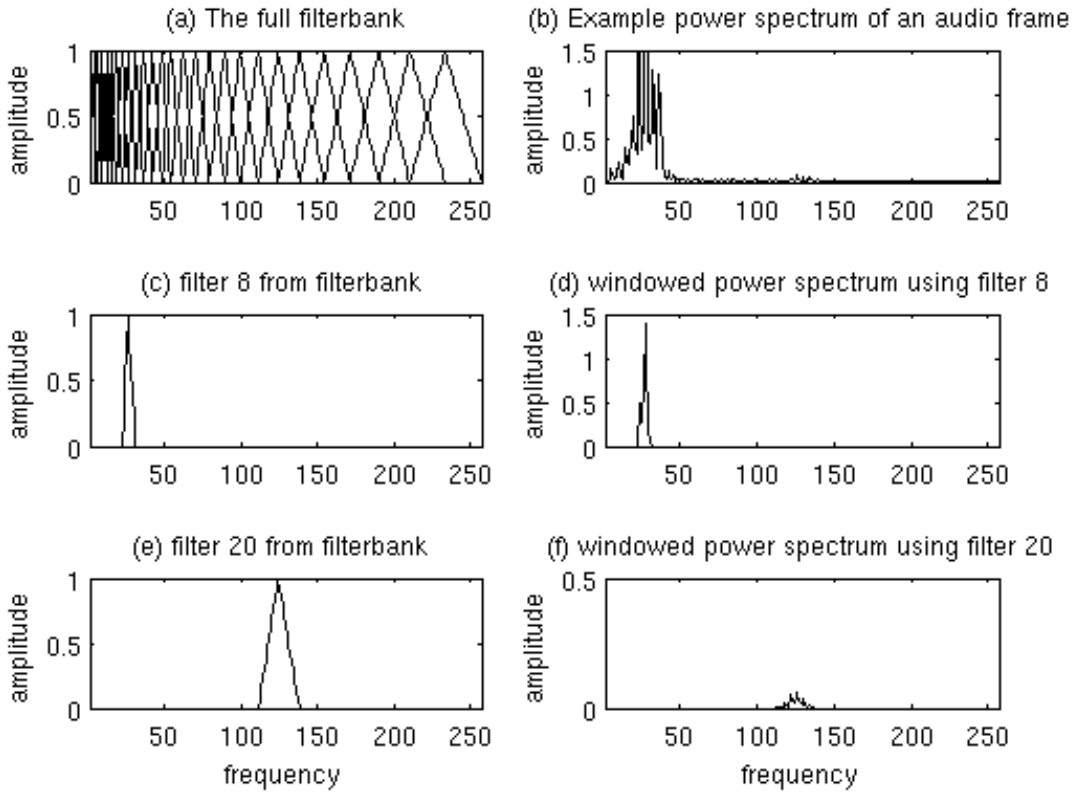

Рисунок 2.2 Мел-кепстральная шкала и энергетический спектр окна

Теперь рассмотрим подробнее способ вычисления фильтров. Для получения фильтров изображенных на рисунке 2.2(a) необходимо выбрать самую низкую и самую высокую частоты. Хорошими значениями являются 300 Гц для нижней частоты и 8000 Гц для высшей частоты. Последующие действия можно описать следующими шагами:

1. Преобразование нижней и верхних частот в мел-шкалу с помощью формулы (2.1). В нашем случаем 300 Гц это 401,25 Мел, 8000 Гц это 2834,99 Мел.

2. Выбор дополнительных точек в мел диапазоне для расчета всех фильтров. Например, для 10 фильтров необходимо выбрать 12 точек. Это означает, что мы должны расположить равномерно 10 точек в диапазоне от 401,25 до 2834,99 Мел. В итоге получим следующие точки:

m(i) = 401,25; 622,50; 843,75; 1065,00; 1286,25; 1507,50; 1728,74; 1949,99; 2171,24; 2392,49; 2613,74; 2834,99.

3. Перевод точек из мел-шкалы в фактические частоты:

h(i) = 300; 517,33; 781,90; 1103,97; 1496,04; 1973,32; 2554,33; 3261,62; 4122,63; 5170,76; 6446,70; 8000.

4. Нахождение точек периодограммы, соответствующих нашим частотам. Каждой точке в периодеграммы соответствует определенная чатота, которая вычисляется по формуле (2.5).

$$f(i) = \frac{2 \cdot (i+1)}{sampleRate \cdot n} \tag{2.5}$$

где *sampleRate* — частота дискретизации сигнала,

n — длина сигнала.

После вычисления чатоты для каждой точки периодограммы находим точки, которые ближе всеге расположенны к точкам, найденным в п.3. В нашем случаем это точки с индексом:

i = 9, 16, 25, 35, 47, 63, 81, 104, 132, 165, 206, 256.

5. Расчет фильров. Первый фильр начинается в первой точке, достигает пика во второй точке, затем возвращается к нулю в третей точке. Второй фильтр начинается во второй точке, достигает максимума в третьей точке и возвращается в нуль в четвертой точке и т. д. Фильтры расчитываются при помощи кусочной функции (2.6).

$$H_m(k) = \begin{cases} 0 & k < f(m-1) \\ \dfrac{k - f(m-1)}{f(m) - f(m-1)} & f(m-1) \le k \le f(m) \\ \dfrac{f(m+1) - k}{f(m+1) - f(m)} & f(m) \le k \le f(m+1) \\ 0 & k > f(m+1) \end{cases} \tag{2.6}$$

где M — число фильтров,

f(i) — частоты периодограммы, соответствующие точкам фильтров (см. п.3).

Результат расчета десяти фильтров представлен на рисунке 2.3.

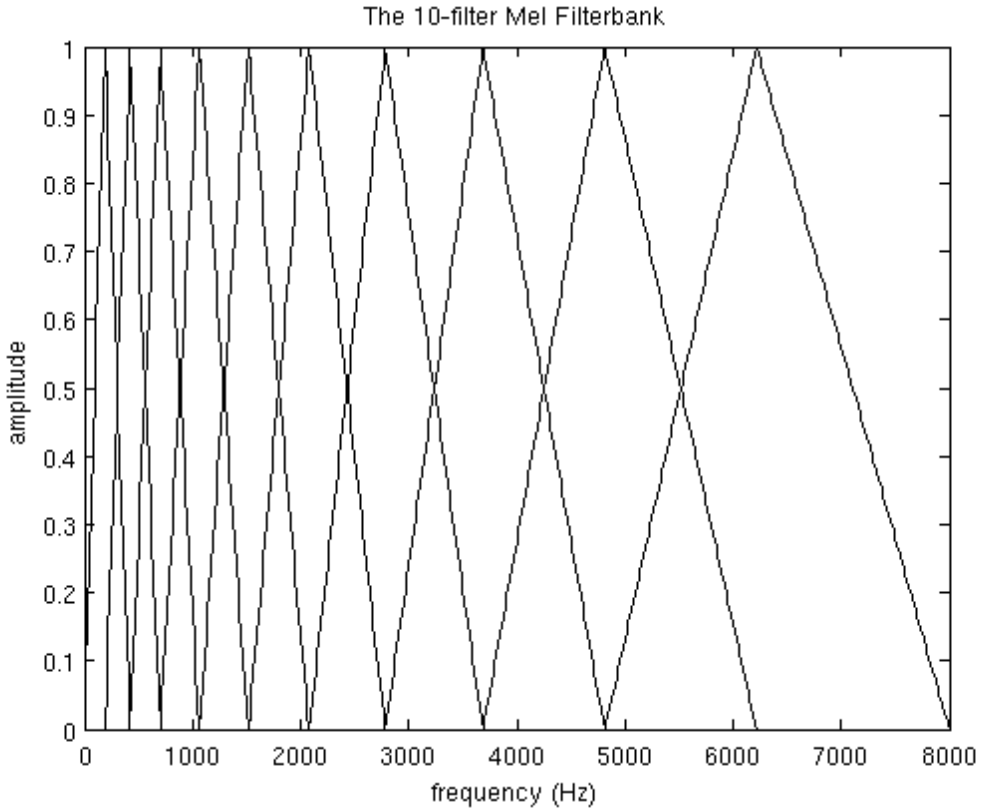

Рисунок 2.3 Пример расчета десяти фильтров [7]

2.2 Метод главных компонент

Однако, как правило, извлеченные признаки объекта также несут в себе некоторую долю избыточной информации за счет корреляции признаков друг с другом. В связи с этим целесообразно выполнить некоторое преобразование полученных признаков, которое снизит корреляции в векторе признаков и уменьшит размерность его пространства. Для этой цели предлагается использовать метод главных компонент (в теории информации он называется преобразование Карунена-Лоева), который позволяет максимизировать скорость уменьшения дисперсии в векторе данных.

Метод главных компонент нацелен на снижение размерности пространства признаков. Теоретически пространство признаков имеет ту же размерность, что и исходное пространство. Однако обычно преобразования выполняются таким образом, чтобы пространство данных могло быть представлено сокращенным количеством "эффективных" признаков. Таким образом, остается только существенная часть информации, содержащейся в

данных. Другими словами, множество данных подвергается сокращению размерности (dimensionality reduction). Для большей конкретизации предположим, что существует некоторый вектор **x** размерности m, который мы хотим передать с помощью l чисел, где $l < m$. Если мы просто обрежем вектор **x**, это приведет к тому, что среднеквадратическая ошибка будет равна сумме дисперсий элементов, "вырезанных" из вектора x. Поэтому возникает вопрос: "Существует ли такое обратимое линейное преобразование **T**, для которого обрезание вектора **Tx** будет оптимальным в смысле среднеквадратической ошибки?" Естественно, при этом преобразование **T** должно иметь свойство маленькой дисперсии своих отдельных компонентов.

Пусть **X** – m-мерный случайный вектор, представляющий интересующую нас среду, E – оператор статистического ожидания, **q** – единичный вектор размерности m, на который проектируется вектор **X**. Это проекция определяется как скалярное произведение **q** на **X**:

$$\mathbf{A} = \mathbf{X}^{\mathrm{T}}\mathbf{q} = \mathbf{X}\mathbf{q}^{\mathrm{T}} \tag{2.7}$$

Матрица R размерности $m \times m$ является матрицей корреляции случайного вектора **X**, которая определяется как ожидание произведения случайного вектора **X** на самого себя:

$$\mathbf{R} = E[\mathbf{X}\mathbf{X}^{\mathrm{T}}] \tag{2.8}$$

Для нахождения векторов **q** необходимо и достаточно выполнения условия (подробное объяснение этого условия приведено в [8]):

$$\mathbf{R}\mathbf{q} = \lambda\mathbf{q} \tag{2.9}$$

В (2.9) легко узнать задачу определения собственных значений. Эта задача имеет нетривиальное решение (т.е. $\mathbf{q} \neq 0$) только для некоторых значений λ, которые называются собственными значениями матрицы корреляции **R**. При этом соответствующие векторы **q** называются собственными векторами. Матрица корреляции характеризуется действительными, неотрицательными собственными значениями. Соответствующие собственные векторы являются единичными. Обозначим собственные значения матрицы **R** размерности $m \times m$ как $\lambda_1, \lambda_2, \ldots \lambda_m$, а

соответствующие им собственные векторы – $\mathbf{q}_1$, $\mathbf{q}_2$, … , $\mathbf{q}_m$ соответственно. Тогда можно записать следующее:

$$\mathbf{R}\mathbf{q}_j = \lambda_j \mathbf{q}, j = 1, 2, \dots, m \quad (2.10)$$

Пусть соответствующие собственные значения упорядочены следующим образом:

$$\lambda_1 > \lambda_2 > \dots > \lambda_j > \lambda_m$$

При этом λ_1 будет равно λ_{max}. Пусть из соответствующих собственных векторов построена следующая матрица размерности $m \times m$:

$$\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_j, \dots, \mathbf{q}_m] \quad (2.11)$$

Тогда систему m уравнений можно объединить в одно матричное уравнение:

$$\mathbf{RQ} = \mathbf{Q}\lambda \quad (2.12)$$

Пусть *вектор данных* **x** является реализацией случайного вектора **X**.

При наличии m возможных значений единичного вектора **q** следует рассмотреть m возможных проекций вектора данных **x**. В частности согласно формуле (2.7):

$$a_j = \mathbf{q}_j^{\mathrm{T}}\mathbf{x} = \mathbf{x}^{\mathrm{T}}\mathbf{q}_j, j = 1, 2, \dots, m \quad (2.13)$$

где a_j – проекции вектора **x** на основные направления, представленные единичными векторами $\mathbf{q}_j$. Эти проекции a_j называются *главными компонентами*. При этом формулы (2.13) можно рассматривать как процедуру анализа.

Для того чтобы восстановить исходный вектор данных **x** необходимо выполнить обратное преобразование:

$$\mathbf{x} = \mathbf{Q}\mathbf{a} = \sum_{j=1}^{m} a_j \mathbf{q}_j \quad (2.14)$$

где $\mathbf{a} = [a_1, a_2, \dots, a_m]^{\mathrm{T}}$. Формулу (2.14) можно рассматривать как процедуру *синтеза*.

С точки зрения задачи статистического распознавания практическое значение анализа главных компонент состоит в том, что он обеспечивает

эффективный способ *сокращения размерности*. В частности, можно сократить количество признаков, необходимых для эффективного представления данных, устраняя те линейные комбинации, которые имеют малые дисперсии, и оставляя те, дисперсии которых велики. Пусть λ_1, λ_2, …, λ_l – наибольшие l собственных значений матрицы корреляции **R**. Тогда вектор данных можно аппроксимировать, отсекая члены разложения (2.14) после l-слагаемого:

$$\hat{\mathbf{x}} = \sum_{j=1}^{l} a_j \, \mathbf{q}_j, l \leq m \tag{2.15}$$

Числа λ_{l+1}, … , λ_m собственными значениями матрицы корреляции **R**. Они соответствуют слагаемым, исключаемым из разложения, чем ближе эти собственные значения к нулю, тем более эффективным будет сокращение размерности в представлении информации исходных данных. Таким образом, для того чтобы обеспечить сокращение размерности входных данных, нужно вычислить собственные значения и собственные векторы матрицы корреляции входных данных, а затем ортогонально проектировать эти данные на подпространство, задаваемых собственными векторами, соответствующими доминирующим собственным значениями этой матрицы [8].

2.3 Искусственные нейронные сети

Для решения задачи распознавания предлагается использовать искусственные нейронные сети (ИНС). Ниже представлены характеристики ИНС, на основе которых было принято решение их использования в качестве классификатора:

1. Хорошая обобщающая способность . ИНС способны к обобщению и показывают хорошие результаты на данных со сложными закономерностями, что является несомненным преимуществом для рассматриваемой задачи, в сравнении с другими классификаторами.

2. Высокая скорость классификации. ИНС работающая в режиме классификации (после завершения стадии обучения), работает практически

мгновенно, так как тратятся ресурсы только для вычисления выходов нейронов с применением ранее полученных нейронных весов.

3. Возможность масштабирования. При добавлении новых данных или классов не требуется вносить изменения в алгоритм работы нейронной сети, достаточно изменить число входных/выходных нейронов и выполнить повторное обучение сети.

4. Адаптивность. Адаптивность достигается за счет адаптивности ее структурных элементов — нейронов, то есть их способности подстраивать свои параметры в процессе изменений условий внешней среды.

Искусственные нейронные сети появились в результате применения математического аппарата к исследованию функционирования нервной системы. Полученные при этом результаты успешно применяются при решении проблем распознавания образов, моделирования, прогнозирования, оптимизации и управления.

Основной структурной и функциональной частью нейронной сети является формальный нейрон (formal neuron), представленный на рис. 6.1, где x_0 , x_1,..., x_n – компоненты вектора входных сигналов, w_0 ,w_1,...,w_n – значения весов входных сигналов нейрона, а y – выходной сигнал нейрона.

Полученные при этом результаты успешно применяются при решении проблем распознавания образов, моделирования, прогнозирования, оптимизации и управления. Основной структурной и функциональной частью нейронной сети является формальный нейрон (formal neuron), представленный на рис. 1, где x_0 , x_1,..., x_n – компоненты вектора входных сигналов, w_0 ,w_1,...,w_n – значения весов входных сигналов нейрона, а y – выходной сигнал нейрона.

Формальный нейрон состоит из элементов 3 типов: умножителей (синапсов), сумматора и преобразователя. Синапс характеризует силу (вес) связи между двумя нейронами. Сумматор выполняет сложение входных сигналов, предварительно помноженных на соответствующие веса. Преобразователь реализует функцию одного аргумента – выхода сумматора. Эта функция называется функцией активации или передаточной функцией

нейрона. Исходя из данного описания, математическая модель нейрона может быть представлена следующим образом:

$$y = f(s) \tag{2.16}$$

$$S = \sum_{i=1}^{n} w_i x_i + b \tag{2.17}$$

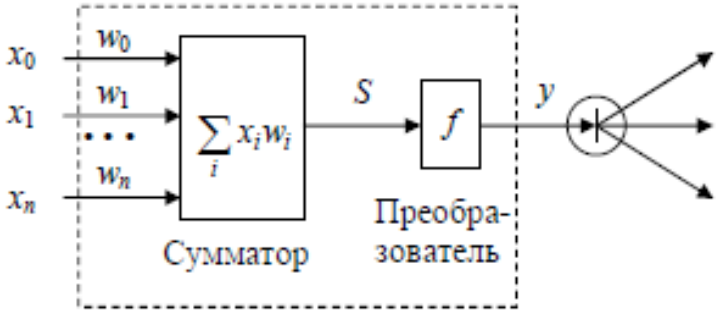

Рисунок 2.4 Формальный нейрон

Примеры некоторых активационных функций представлены в таблице 2.1.

Таблица 2.1 — Функции активации нейронов

Название	Формула	Область значений
Пороговая	$f(S) = \begin{cases} 0, & S < \Theta \\ 1, & S \geq \Theta \end{cases}$	(0;1)
Линейная	$f(S) = aS$	(-∞; +∞)
Лог-сигмоидная	$f(S) = \frac{1}{1 + e^{-aS}}$	(0;1)
Гиперболический тангенс	$f(S) = \frac{e^{aS} - e^{-aS}}{e^{aS} + e^{-aS}}$	(-1;1)

Описанные формальные нейроны можно объединять таким образом, что выходные сигналы одних нейронов являются входными для других. Полученное множество связанных между собой нейронов называют искусственными нейронными сетями (artificial neural networks, ANN) или, коротко, нейронными сетями [8].

На рисунке 2.5 представлена искусственная нейронная сеть прямого распространения с двумя входными и выходными нейронами и одним скрытым слоем с 3-мя нейронами.

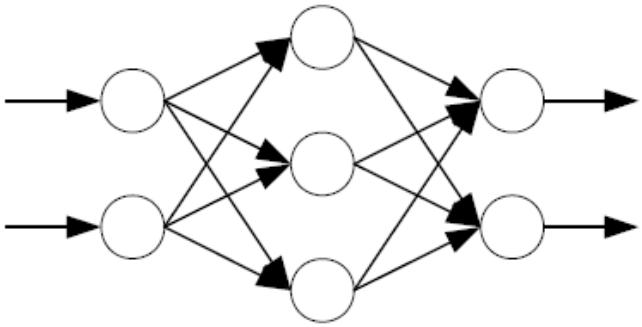

Рисунок 2.5. Искусственная нейронная сеть прямого распространения

2.4 Метод обратного распространения ошибки

Наиболее распространённым методом обучения нейронный сетей прямого распространения является метод обратного распространения ошибки (backpropagation method). Данный алгоритм используется для минимизации отклонения выходных сигналов сети от желаемых.

Примем в качестве функции ошибки следующую величину:

$$E(w) = \frac{1}{2}\sum_{i,k}(f_{i,k} - y_{i,k}^{(T)})^2, \tag{2.18}$$

где $f_{i,k}$ – значение выходного сигнала k-го выходного нейрона при подаче на вход i-го набора обучающих данных, $y_{i,k}^{(T)}$ –требуемое значение k-го нейрона для i-го набора данных из обучающей выборки. Обучение нейронной сети нацелено на минимизацию функции $E(w)$.

Минимизация методом градиентного спуска обеспечивает подстройку весов следующим образом:

$$\Delta w_{ij}^{(q)} = \eta \frac{\partial E}{\partial w_{ij}}, \tag{2.19}$$

где $\Delta w_{ij}^{(q)}$ – величина изменения веса связи, соединяющей i-й нейрон (q–1) слоя с j-м нейроном слоя q; η – коэффициент скорости обучения, $0 < \eta < 1$. Таким образом, вес связи изменяется пропорционально ее вкладу в значение ошибки

нейрона, для которого эта связь является входной, так как частная производная по весу $\frac{\partial E}{\partial w_{ij}}$ показывает зависимость скорости изменения функции ошибки E от изменения этого веса.

Изменения веса определяется следующим образом:

$$\Delta w_{ij}^{(q)} = -\eta \delta_i x_i, \tag{2.20}$$

где δ_i – значение ошибки j-го нейрона в слое q, x_i – значение i-го входного сигнала для j-го нейрона q слоя.

Значение ошибки нейронов выходного слоя:

$$\delta_i = \left(f_{i,k}(S)\right)' \left(f_{i,k} - y_{i,k}\right), \tag{2.21}$$

где $y_{i,k}$ – требуемое, а $f_{i,k}$ – фактическое значение выходного сигнала k-го нейрона для i-го набора данных из обучающей выборки, $(f_{i,k}(S))'$ – значение производной активационной функции k-го нейрона для i-го набора обучающих данных.

Если нейрон принадлежит одному из скрытых слоев, то

$$\delta_i^{(q)} = (f_i^{(q)}(S))' \sum_j w_{ij} \delta_j^{(q+1)} \tag{2.22}$$

где $\delta_i^{(q)}$ – ошибка i-го нейрона в слое q, $\delta_j^{(q+1)}$ – ошибка j-го нейрона в слое $(q+1)$, w_{ij} – вес связи, соединяющей эти нейроны, $(f_{i,k}(S))'$ – значение производной активационной функции i-го нейрона слоя q. Таким образом, значение ошибки нейрона пропорционально его влиянию на величины ошибок нейронов следующего слоя, а также скорости изменения его выходного сигнала для k-го набора обучающих данных.

Для нейронных сетей с лог-сигмоидными функциями активации:

$$f(S) = \frac{1}{1 + e^{-aS}}, \tag{2.23}$$

где a – константа, S – взвешенная сумма входных сигналов нейрона, тогда

$$f'(S) = af(S)\big(1 - f(S)\big), \tag{2.24}$$

формулы (2.21), (2.22) примут вид

$$\delta_i = af_{i,k}\left(1 - f_{i,k}\right)\left(f_{i,k} - fy_{i,k}\right), \quad (2.25)$$

$$\delta_i^{(q)} = af_i(1 - f_i)\sum_j w_{i,j}\,\delta_j^{(q+1)}, \quad (2.26)$$

Для реализации алгоритма обратного распространения ошибки может быть использована следующая последовательность действий:

1. Предъявление очередного набора из обучающей выборки на вход нейронной сети.

2. Вычисление выходного сигнала сети.

3. Определение величин ошибок нейронов выходного слоя по формуле (2.21) или (2.25).

4. Определение величин ошибок нейронов скрытых слоев по формулам (2.22) или (2.26).

5. Однократная коррекция весов связей.

6. Если в обучающей выборке есть неиспользованные в данной эпохе наборы данных, то переход на шаг 1.

7. Подсчет ошибки сети по формуле (2.18). Если ошибка меньше заданной, то конец обучения, иначе, начало новой эпохи обучения и переход на шаг 1 [8].

2.5 Нейроэволюционный алгоритм Enforced Subpopulations

Нейроэвллюционные алгоритмы предназначены для подбора весов нейронов и архитектуры сети посредством применения эволюционных (генетических) алгоритмов. Информация о весах и структуре сети кодируется в хромосомах, затем происходят стандартные этапы работы генетического алгоритма:

1. Скрещивание
2. Мутация
3. Оценка приспособленности
4. Отбор

Кодирование информации в хромосомах может являться прямым и косвенным. Во время использования прямого кодирования информация о всех весах и структуре записывается в хромосому в исходном виде, при косвенном кодировании – требуется осуществить некоторое преобразования генотипа (внутренней генетической информации) в фенотип (внешние признаки присущие индивиду).

Нейроэволюционные алгоритмы позволяют обучать нейронные сети в случае, когда неизвестен желаемый отклик сети, а лишь можно оценить относительное качество результата (лучше, хуже). Но также они пригодны для задач с известным желаемым откликом. Например, их можно использовать при наличии множества маркированных примеров (обучающей выборки). Одним из преимуществ генетических алгоритмов, например в сравнении, с методом обратного распространения ошибки является то, что оценивается полностью решение (вся ИНС), а не происходит подстройка под конкретные входные-выходные данные, что теоретически может улучшить способность нейронной сети к обобщения. Кроме того, некоторые из нейроэволюционных алгоритмов позволяют подбирать оптимальную структуру сети, что является очень важным, так как подбор архитектуры сети вручную требует сравнительно больших трудозатрат и не всегда удается подобрать ее оптимальную структуру.

Алгоритм Enforced Subpopulations (ESP) предназначен для подбора весов нейронов и оптимальной структуры искусственной нейронной сети. Он был предложен Фаустино Гомесом [9].

Может быть использован для обучения любой сети с одним скрытым слоем:

- Прямого распространения
- Сети Элмана
- Полносвязной рекуррентной

Используется прямое вещественное кодирование. Хромосома содержит следующие параметры:

- Веса всех входных связей

- Веса всех выходных связей
- Веса всех рекуррентных связей

Общая схема алгоритма может быть описана следующим образом:

1. Инициализация
   a. Задается *h* скрытых нейронов
   b. Создается *h* подпопуляций из *n* особей
2. Оценка приспособленности
   a. Каждый нейрон принимает участие в попытках (*trials*), при которых он помещается в нейронную сеть, случайно сформированную из нейронов других популяций
   b. Приспособленность каждой особи (нейрона) суммируется (*кумулятивная приспособленность*)
   c. Оценивание производиться до тех пор, пока каждый нейрон не будет использован как минимум в 10 попытках
3. Проверка вырождения популяции
   a. Если приспособленность лучшей особи не улучшается в течение последних *b* поколений, то производится взрывая мутация (*burst mutation*)
   b. Если после двух взрывных мутаций нет улучшения, то изменяется структура ИНС
4. Скрещивание
   a. Вычисление средней приспособленности для каждого нейрона: кумулятивная приспособленность делится на число попыток
   b. Сортировка нейронов в каждой популяции в порядке ухудшения приспособленности
   c. Из популяции удаляются лишние особи, т.е. номер которых превосходит начальный размер популяции
   d. Скрещивается 1/4 часть лучших особей подпопуляции. Используется одноточечный кроссинговер. Потомки добавляются в конец подпопуляции.

e. Для нижней половины производится мутация с распределением Коши

Шаги 2-4 повторится до тех пор, пока не будет достигнут критерий остановки.

Алгоритм взрывной мутации может быть описан следующим образом:

1. Для каждой подпопуляции определяться наилучшая особь
2. Генерируются новые подпопуляции вокруг своих лучших особей
3. Используется распределение Коши

$$f(x) = \frac{\alpha}{\pi(\alpha^2 + x^2)} \tag{2.18}$$

Алгоритм адаптации структуры ИНС:

1. Уменьшение размера популяции
   a. По очереди отключается одна подпопуляция. Оставшиеся *h–1* подпопуляций оцениваются стандартным образом.
   b. Если лучшая приспособленность новой ИНС, полученной после отключения подпопуляции, лучше, чем лучшая приспособленность до отключения, то подпопуляции удаляется.
2. Если ни одна подпопуляция не была удалена, то к популяции добавляется новая подпопуляция со случайно проинициализированными нейронами.

В результате процесса эволюции происходит специализация нейронов. Она заключается в том, что работа нейрона зависит от занимаемого положения в сети, поэтому нейроны должны иметь разные веса и решать разные задачи. На рисунке 2.6 изображена проекция весов на плоскость первых двух собственных векторов в начале поиска, а также после 20, 50 и 100 поколений. Разными цветами выделены веса разных нейронов.

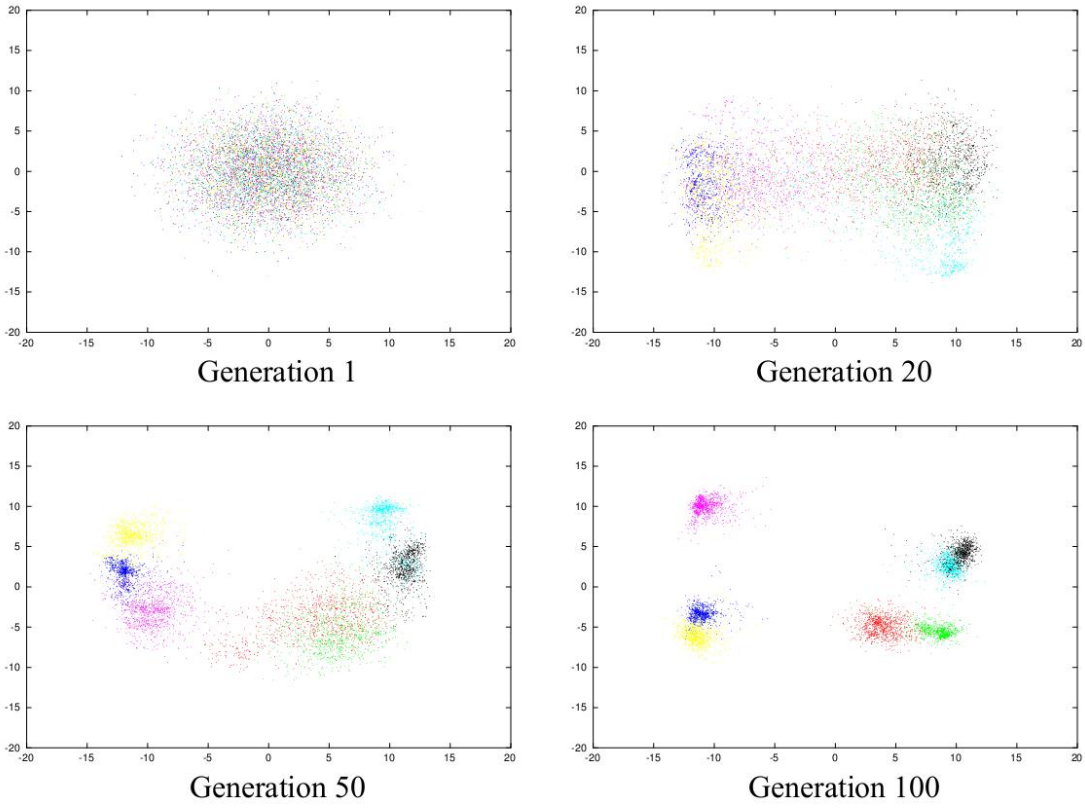

Рисунок 2.6 – Иллюстрация специализации нейронов [9]

2.6 Алгоритм распознавания музыкальных инструментов

В соответствие с выбранным подходом, алгоритм распознавания тембра музыкального инструмента можно описать следующий образом:

1. Этап сегментации, который подразумевает разбиение входного сигнала на интервалы. Целесообразно разбивать сигнал на небольшие промежутки от 20 до 100 мс, так как сигнал изменяется во времени и использование больших интервалов может привести к снижению качества распознавания. В данной работе был выбран сравнительно большой интервал ~100 мс (4096 отсчетов при частоте дискретизации 44 100 Гц), для ускорения скорости обработки и снижения числа входных данных, так как на текущем этапе данная работа носит исследовательский, нежели прикладной характер.

2. Следующим этапом является этап предобработки, который включает в себя применение оконной функции Хэмминга для каждого сегмента для подавления высоких частот на границах сегмента. Данные частоты являются результатом обрезания сигнала и негативно сказывается на точности распознавания.

3. Вычисление преобразования Фурье для каждого входного интервала. В данной работе было использовано быстрое преобразование Фурье (Fast Fourier Transformation).

4. Вычисление MFCC.

5. Снижение размерности пространства признаков посредством применения метода главных компонент.

6. Обучение нейронной сети на основе полученных коэффициентов. Для обучения нейронной сети был использован метод градиентного спуска, а именно метод обратного распространения ошибки (backpropogation).

7. Дополнительным модулем в структуре программы является модуль нейроэволюционного алгоритма, которые заменяет собой метод обратного распространения ошибки и позволяет обучать нейронную сеть при помощи эволюции нейронов, а также подбирает ее оптимальную структуру.

На следующем рисунке представлена структурная схема алгоритма.

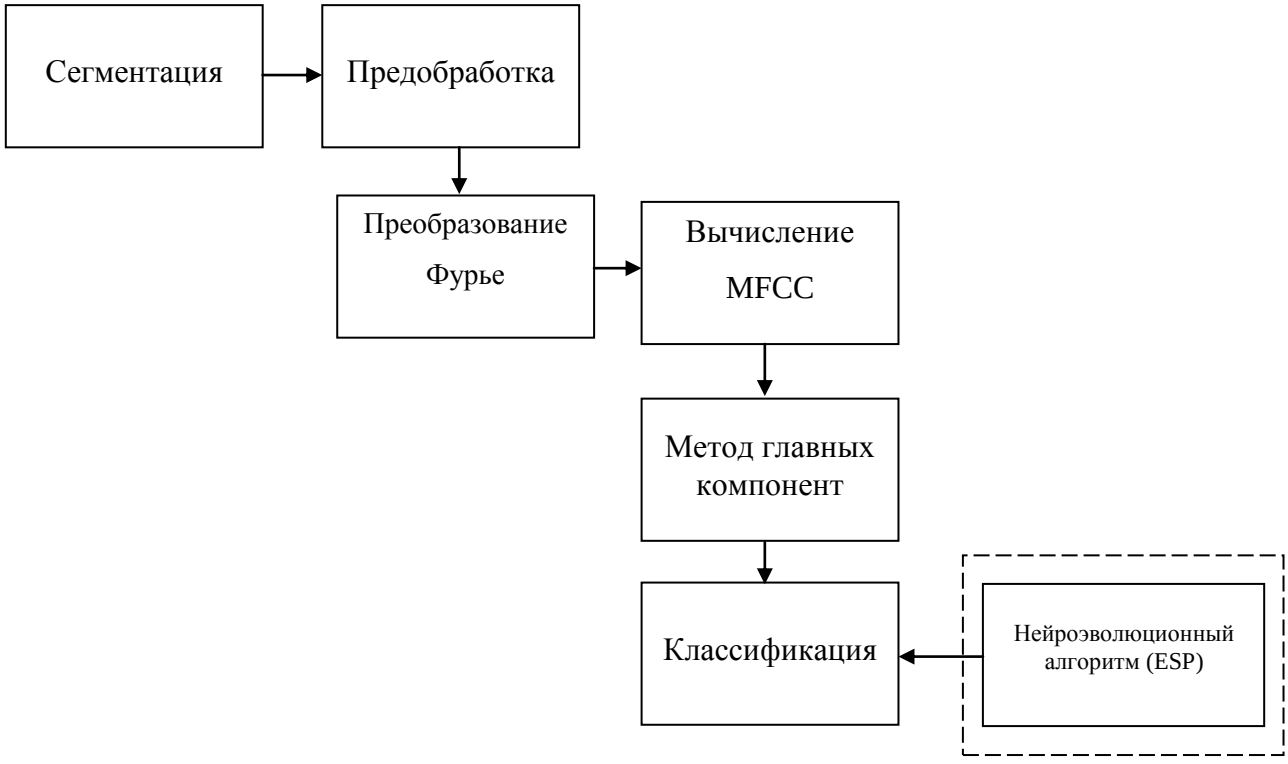

Рисунок 2.7 – Структурная схема алгоритма

3 РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

3.1 Общие сведения

Алгоритм распознавания музыкальных инструментов реализован в виде программы на языке C#, кодовое название программы — Timber Recognition.

Перечень программных модулей приведен в таблице 3.1.

Таблица 3.1 — Программные модули

Название	Назначение
TimbreRecognition.exe	Исполняемый модуль — содержит основную логику программы
Accord.Audio.dll	Модуль библиотеки Accord, предоставляющей возможность чтения файлов различный аудио-форматов

Помимо основной логики, в состав программы также входит модуль библиотеки *Accord.NET*. *Accord.NET* — свободно распространяемая библиотека, реализующая различные математические алгоритмы обработки статистических данных, изображений, а также алгоритмы обработки аудиоданных.

Программа исполняется в операционной среде Windows (начиная с версии XP) при наличии установленного системного пакета — Microsoft Framework версии 4.0.

Данное программное обеспечение разрабатывалось в среде разработки Microsoft Visual Studio 2012.

Среди потенциальных языков для реализации алгоритма рассматривались Java, C++ и C#. В конечном итоге был выбран язык C#, по причине того, что в Java затруднено проектирование графического интерфейса, и требует больших трудозатрат в сравнении с C#, в то время как графический интерфейс пользователя и возможность визуализации результатов являются важными частями приложения. Язык C++ не был выбран для разработки приложения по причине необходимости ручного управления памятью и сложности работы с указателями.

3.2 Функциональное назначение

Программа Timbre Recognition предназначена для обучения и тестирования многослойных нейронных сетей для решения задачи распознавания музыкальных инструментов. Тестирование нейронных сетей может производиться как на сигналах отдельно звучащих музыкальных инструментов, так и на более сложных сигналах — полифонических мелодиях, в которых разработанная программа способна идентифицировать сольные партии отдельных музыкальных инструментов.

Помимо этого программа позволят анализировать музыкальные данные на предмет близости их нахождения в пространстве признаков. Данный анализ полезен во время процедуры отбора признаков для последующего их распознавания. Также реализована визуализация признаков посредством специального типа нейронных сетей — самоорганизующихся карт Кохонена.

3.3 Описание логической структуры

В этом подразделе будет дано описание логической структуры программы. Для этой цели используется унифицированный язык моделирования (UML) версии 2.0. Все диаграммы, приведенные в данном подразделе, выполнены с использованием программного средства StarUML.

Условно программу можно разбить на три модуля: модуль пользовательского интерфейса, модуль извлечения признаков и модуль работы с нейронными сетями. Диаграмма модулей представлена на рисунке 3.1.

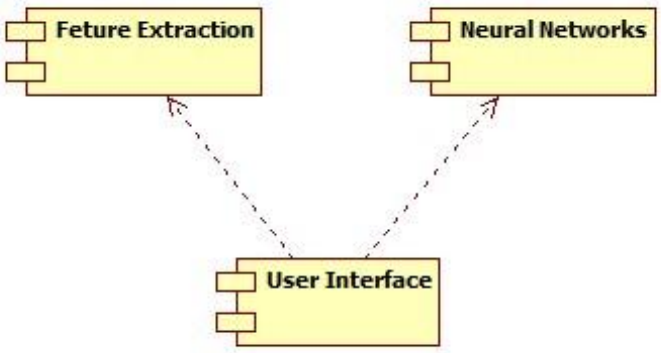

Рисунок 3.1 — Диаграмма модулей программы

Neural Networks — модуль, инкапсулирующий логику по искусственными работе с нейронными сетями (ИНС).

Feature Extraction — модуль, инкапсулирующий в себе логику по извлечению признаков.

User Interface — модуль, содержащий форму оконного пользовательского интерфейса, использующий, соответственно, модули Feature Extraction и Neural Networks.

На следующей диаграмме 3.2 представлены классы-сущности модуля *Neural Networks*. Необходимо отметить, что для бóльшей ясности на диаграммах представлены только базовые поля и методы, необходимые для объяснения основной концепции работы того или иного модуля программы.

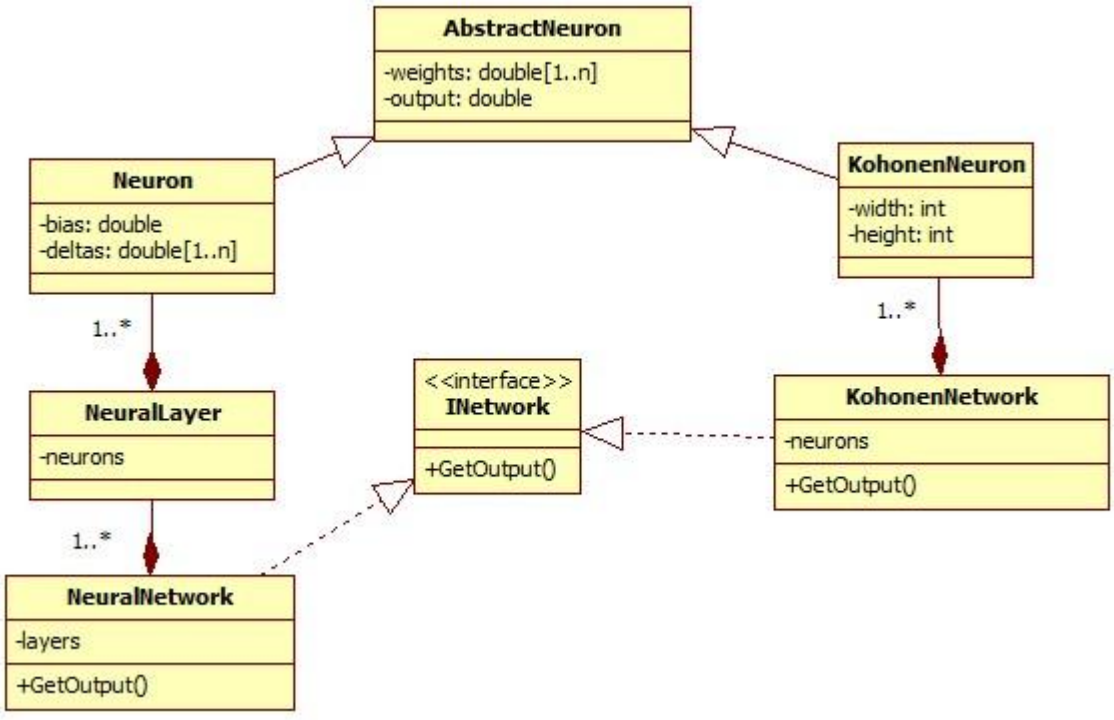

Рисунок 3.2 — Диаграмма классов «Нейронные сети»

Класс *AbstractNeuron* — базовый класс для классов *Neuron* и *Kohoneneuron*, содержащий общие данные: массив весовых коэффициентов и текущее значение выхода нейрона.

Класс *Neuron* (унаследован от класса *AbstractNeuron*)— является базовым структурным элементом классической нейронной сети (многослойного

персептрона). Помимо полей суперкласса содержит такие поля как bias (смещение нейрона) и deltas (используется для обучения с «памятью»). Также в этом классе содержится активационная функция (на диаграмме опущена), может быть линейной, логсигмоидной или гиперболический тангенсом.

Класс *NeuronLayer* представляет собой слой нейронной сети и содержит в себе коллекцию элементов класса *Neuron.*

Класс *NeuralNetwork* представляет собой реализацию многослойного персептрона прямого распространения, содержит в себе коллекцию объектов класса *NeuronLayer.* Реализует интерфейс *INetwork*, предоставляющий метод для получения выходного значения сети.

Класс Kohonen*Neuron* (унаследован от класса *AbstractNeuron*)— является основным структурным элементов самоорганизующихся карт Кохонена, используемых в данном приложении для графического анализа пространственных данных. Помимо полей базового класса содержит такие поля как *width* и *height* — размеры карты, т.е. число нейронов в двух измерениях самоорганизующейся карты.

На диаграмме 3.3 представлены классы, реализующие процесс обучения нейронной сети.

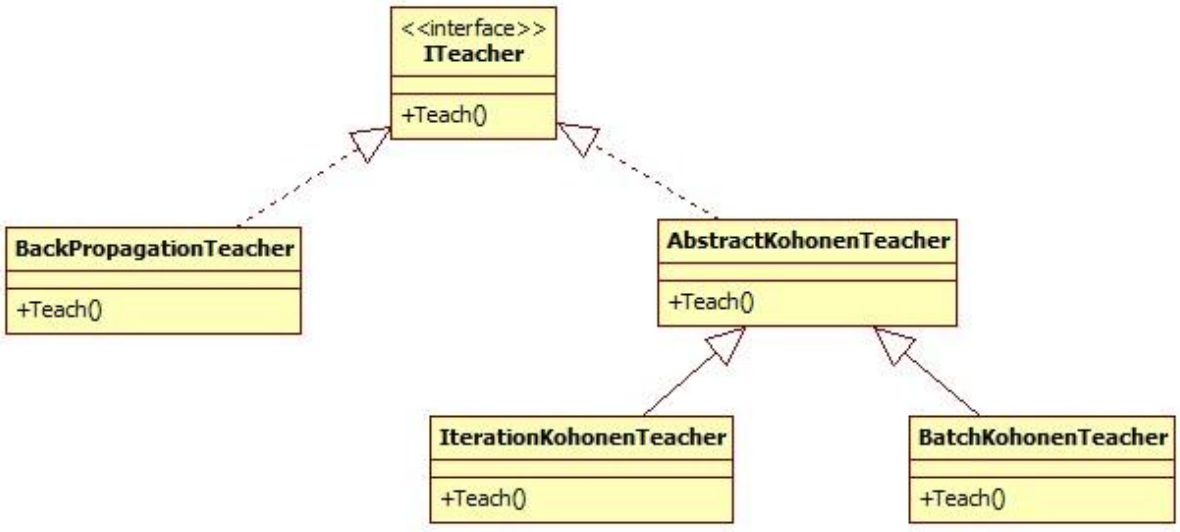

Рисунок 3.3 — Диаграмма классов «Обучение нейронных сетей»

Все классы реализует базовый интерфейс ITeacher, предоставляющий функциональность по обучению нейронных сетей (метод Teach). Данный интерфейс необходим для простой замены одной реализации другой.

Класс *BackgroundPropagationTeacher* реализует функциональность по обучения многослойного персептрона методом обратного распространения ошибки.

Класс *AbstractKohenenTeacher* реализует общую логику по обучению самоорганизующихся карт Коханена. Является базовым для классов *IterationKohenenTeacher* и *BatchKohenenTeacher*.

Класс *IterationKohenenTeacher* реализует логику обучения карт Коанена итеративным методом.

Класс *BatchKohenenTeacher* реализует логику обучения карт Коанена пакетным методом обучения.

На рисунке 3.4 представлена диаграмма классов, описывающая функционал по созданию нейронных сетей.

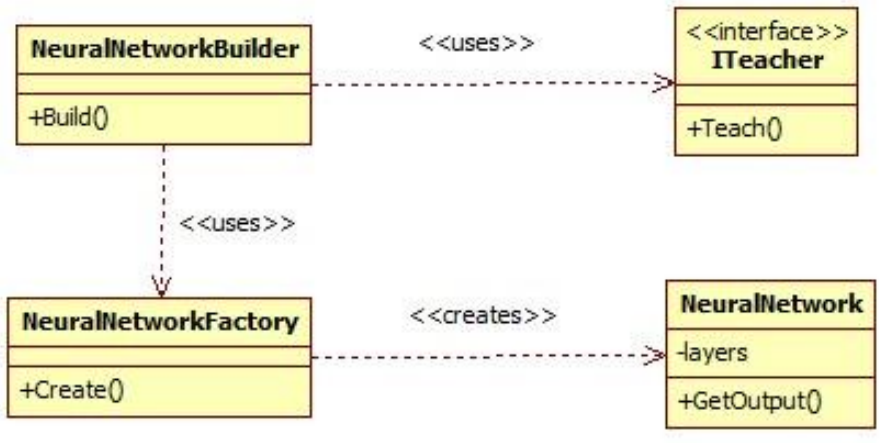

Рисунок 3.4 — Диаграмма классов «Создание нейронных сетей»

За процесс создание и обучения нейронных сетей отвечает класс *NeuralNetworkBuilder*. Этот класс использует *NeuralNetworkFactory* для создания нейронных сетей с требуемой архитектурой. Класс *NeuralNetworkBuilder* инициализируется одной из реализаций интерфейса *ITeacher*, которая в свою очередь используется для обучения нейронной сети.

Класс *NeuralNetworkFactory* предназначен для создания экземпляров класса *NeuralNetwork* и является реализацией шаблона проектирования *Factory Method*. Класс содержит метод *Create,* который принимает в качестве аргументов основные параметры нейронной сети: число входных и выходных нейронов, число нейронов в скрытом слое и число скрытых слоев; а возвращает экземпляр класса *NeuralNetwork* с заданными параметрами.

На рисунке 3.5 изброжена диаграмма последовательностей, иллюстрирующая процесс создания и обучения нейронной сети.

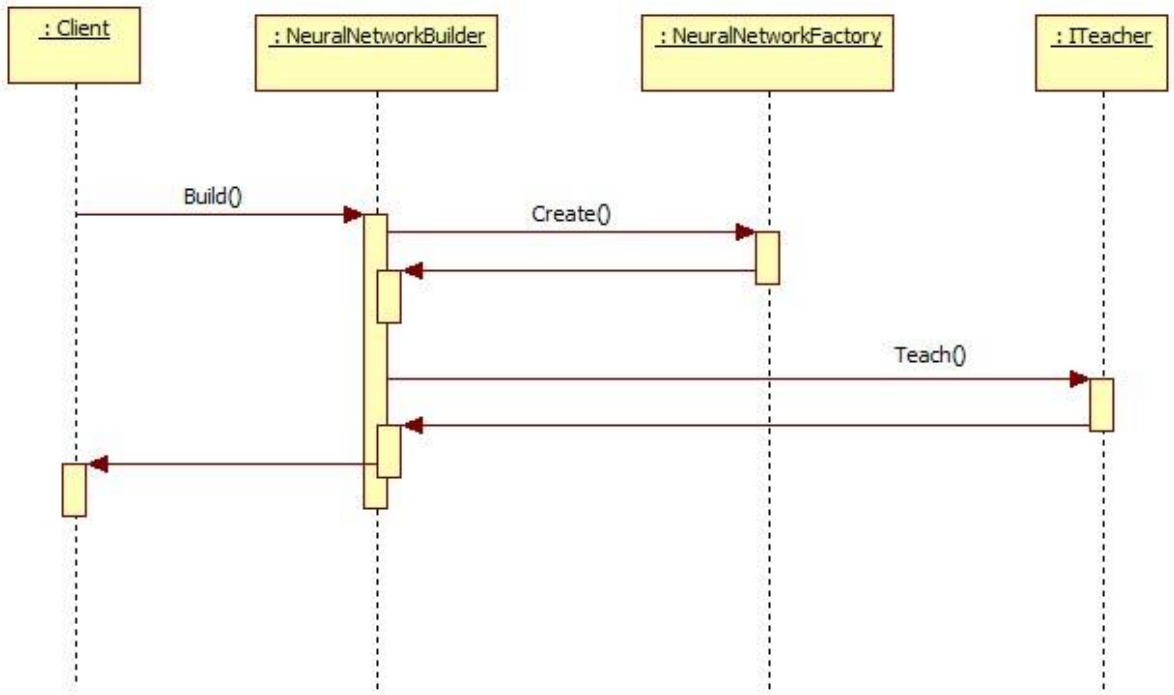

Рисунок 3.5 — Диаграмма последовательностей «Создание и обучение нейронных сетей»

Класс *Client* — некий класс, олицетворяющий код клиентского приложения, например, может быть представлен формой пользовательского интерфейса.

На рисунке 3.6 изображена диаграмма вспомогательных классов, используемых в процессе классификации.

Метод *GetOutput* класса *NeuralNetwork* возвращает выход сети в виде массива чисел с плавающей точкой, однако для решения задачи классификации необходимо определить нейрон, значение которого является максимальным,

это и будет показателем принадлежности входных данных к соответствующему классу. Для этой целей используется класс *NeuralNetworkWrapper*, содержащий метод *Classify*. Данный класс содержит в себе в качестве поля экземпляр класса *NeuralNetwork.*

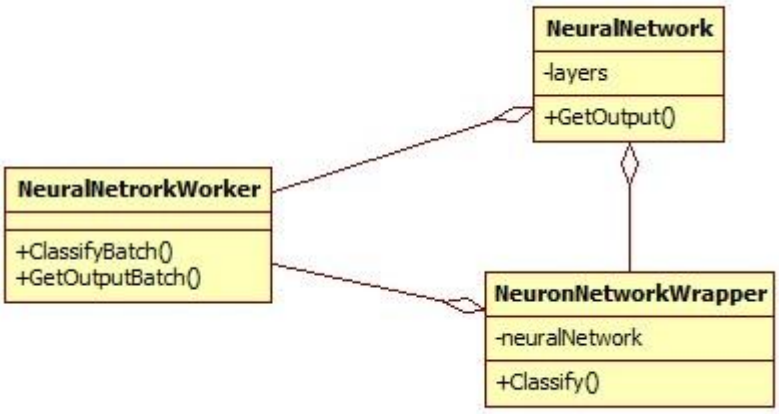

Рисунок 3.6 — Диаграмма классов «Классификация с использованием нейронных сетей»

Класс *NeuralNetworkWorker* инкапсулирует в себе классы *NeuralNetwork* и *NeuralNetworkWrapper* предоставляет интерфейс для массовой оценки и классификации данных.

Теперь перейдем к рассмотрению классов модуля Feature Extraction. На рисунке 3.7 представлена диаграмма базовых классов данного модуля.

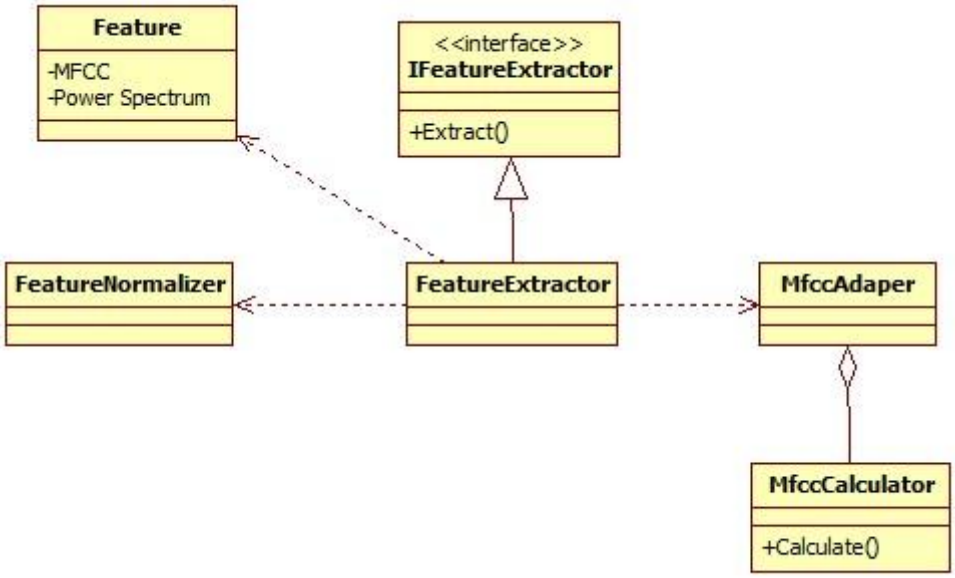

Рисунок 3.7 — Диаграмма классов «Извлечение признаков»

Класс *Feature* инкапсулирует в себе признаки, такие как *MFCC*, *Power Spectrum* и др.

Класс *FeatureExtractor* реализует интерфейс *IExtractor* и выполнят вычисление признаков самостоятельно или делегирую это другим классам, например *MfccAdapter*.

Так как часть кода, вычисляющего MFCC, была заимствована, то было принято решение расположить данный код в отдельном классе (*MFCCCalculator*), а для его совместимости с остальным кодом создать класс *MFCCAdapter*, который занимается преобразованием входных и выходных данных.

Класс *FeatureNormalizer* содержит в себе логику по нормализации полученных признаков.

На рисунке 3.8 приведена диаграмма, на которой изображены классы, необходимые для вычисления MFCC.

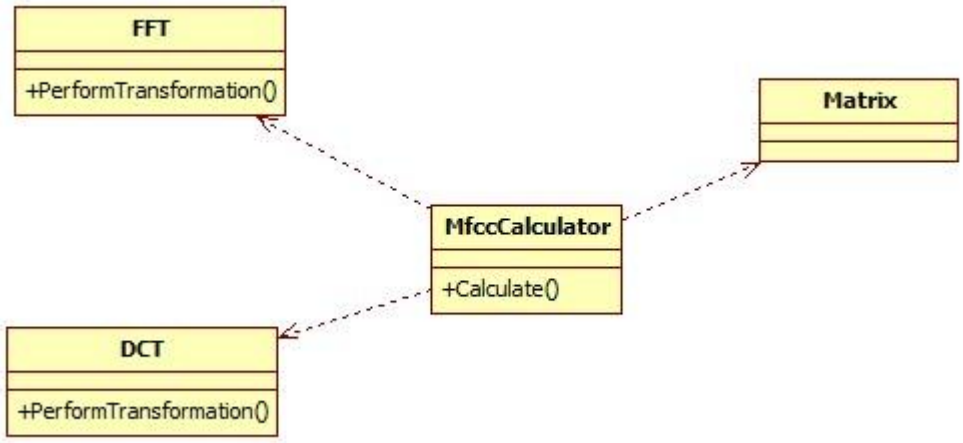

Рисунок 3.8 — Диаграмма классов «Вычисление MFCC»

Класс *FFT* выполняет быстрое дискретное преобразование Фурье (Fast Fourier Transformation).

Класс *DCT* выполняет дискретное косинусное преобразование (Discrete Cosines Transformation).

Класс *Matrix* представляет собой математическую матрицу и содержит реализацию всех необходимых матричных операций.

На рисунке 3.9 приведена диаграмма последовательностей, иллюстрирующая процесс извлечения признаков.

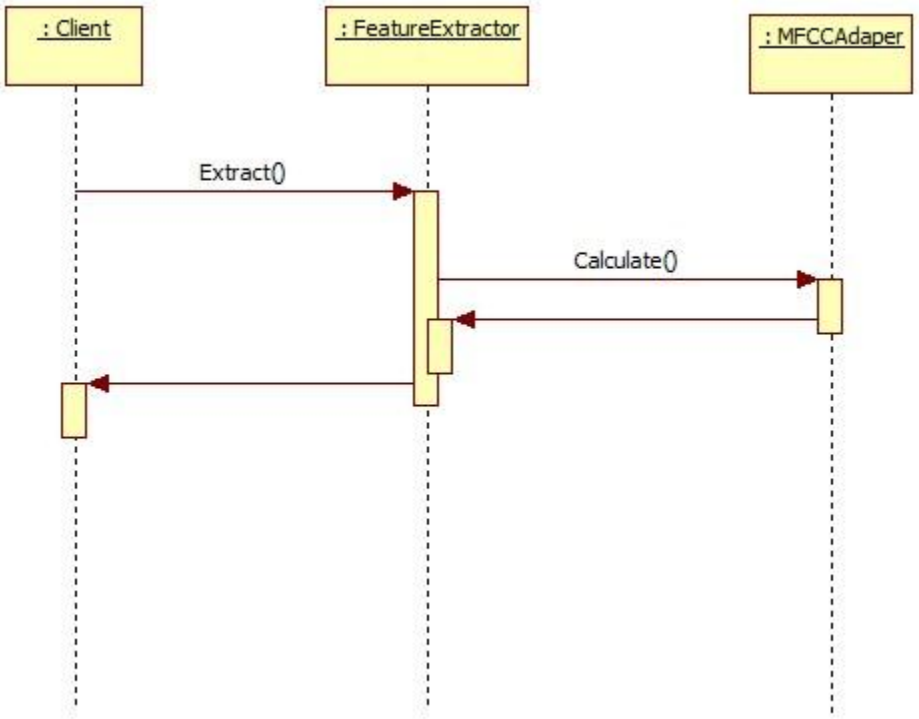

Рисунок 3.9 — Диаграмма последовательностей «Извлечение признаков»

На рисунке 3.10 проиллюстрирован весь процесс распознавания данных в виде диаграммы последовательностей.

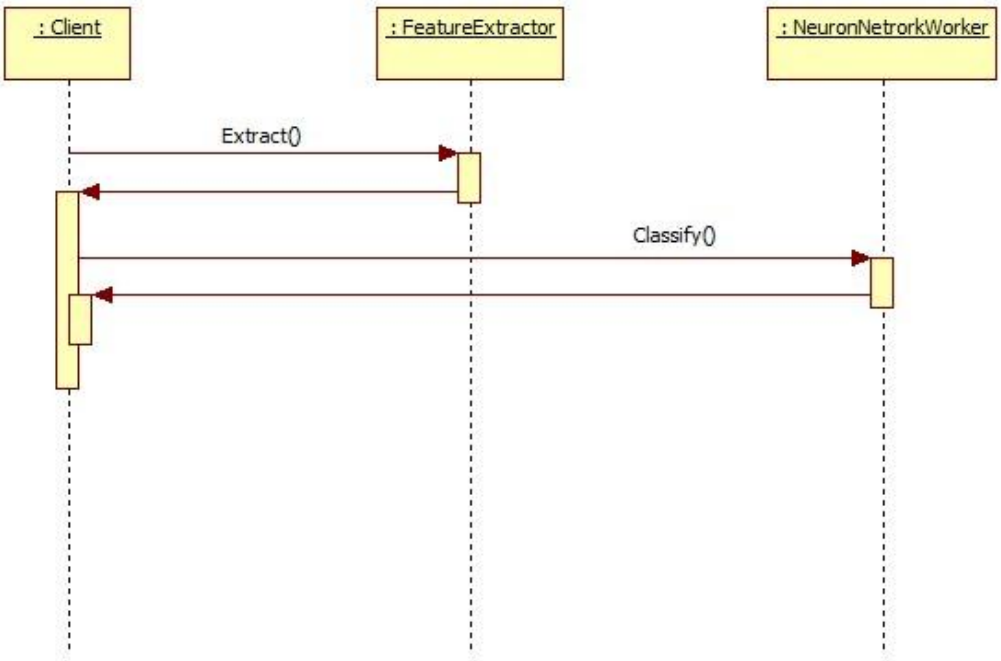

Рисунок 3.10 — Диаграмма последовательностей «Процесс классификации»

3.4 Входные данные

Входными данными являются параметры нейронной сети, звуковые файлы для обучения и тестирования сети. Параметры нейронной сети представляют собой число скрытых слоев и число нейронов в скрытом слое. Звуковые файлы должны быть в формате Waveform Audio File Format (WAV).

3.5 Выходные данные

Выходными данными работы программы является текстовый файл, содержащий веса всех нейронов в виде чисел с плавающей точкой. Также к выходным данным можно отнести результат тестирования сети в текстовом и графическом виде.

3.6 Графический интерфейс пользователя

На рисунке 3.11 представлен внешний вид программы.

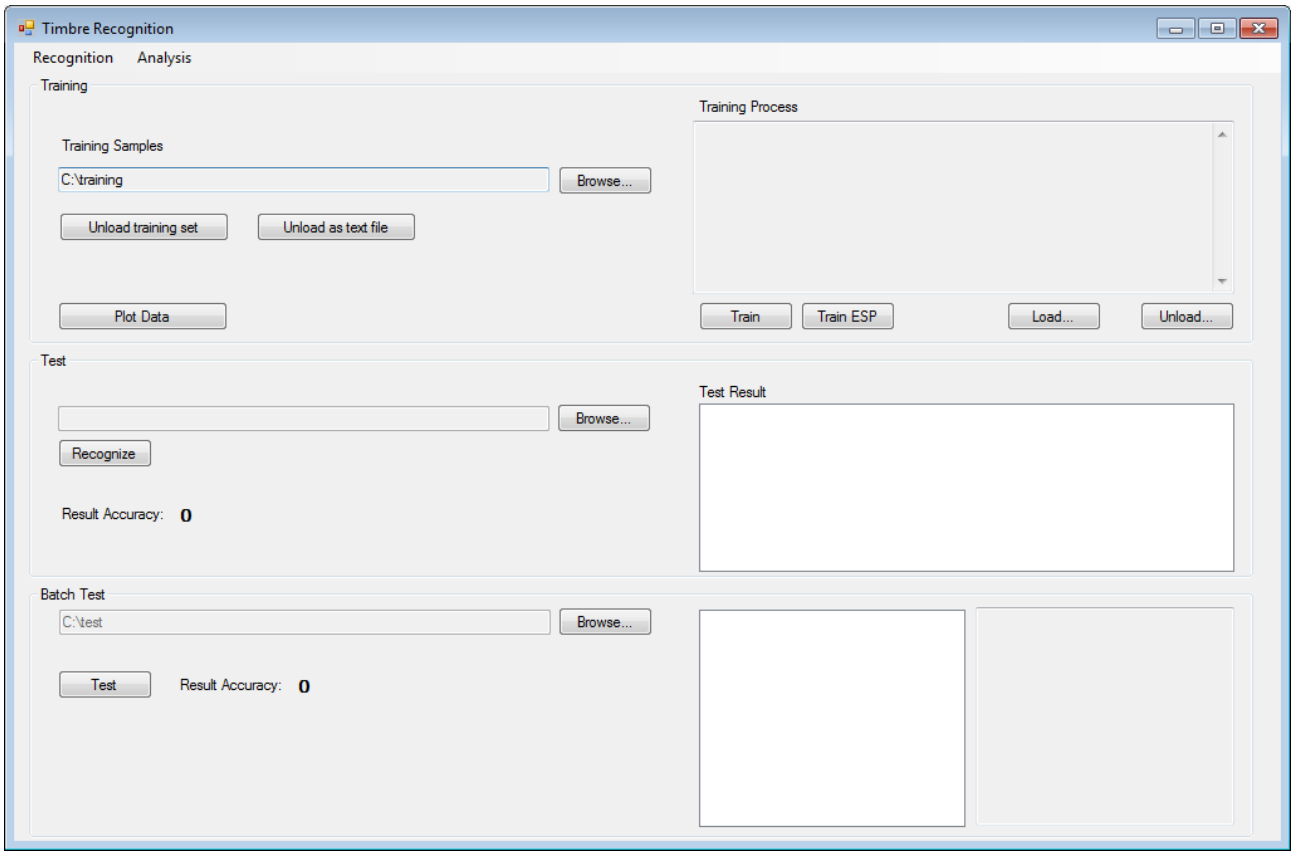

Рисунок 3.11 — Внешний вид программы

Программы состоит из трех основных частей: модуль обучения сети (Training), модуль тестирования (Test), модуль пакетного тестирования (Batch Test).

Модуль обучения позволяет выполнять процесс обучения искусственной нейронной сети (ИНС) в двух режимах: стандартное обучение методом обратного распространения ошибки (кнопка Train) и нейроэволюционным алгоритмом ESP (кнопка Train ESP). Также данный модуль содержит дополнительные возможности по работе с обучающими данными и нейронной сетью: выгрузка признаков в файл, визуализация данных в 3D режиме, загрузка структуры нейронной сети из файла, импорт ИНС в файл.

Модуль тестирования позволяет выполнять тестирования отдельного файла и анализ результатов распознавания.

Модуль пакетного тестирования позволяет выполнять тестирование группы файлов и получать статистические данные по каждому классу.

Дополнительно программа имеет два пункта меню: recognition и analysis. Пункт Recognition позволяет выполнять распознавание отдельного файла в графическом режиме. Пункт Analysis позволят визуально анализировать пространство признаков с помощью самоорганизующихся карт Кохонена.

4 АНАЛИЗ РЕЗУЛЬТАТОВ

4.1 Анализ числа MFCC и точности распознавания

Важным вопросом является определение количества MFCC необходимых для получения высокой точности распознавания музыкальных инструментов при желательной минимизации общего числа признаков.

Для определения минимального необходимого количества коэффициентов был поставлен эксперимент, в котором изменялось число коэффициентов от 1 до 20ти, обучение производилось на пяти инструментах. В ходе эксперимента искусственная нейронная сеть обучалась 10 раз с каждым набором коэффициентов.

Результаты эксперимента изображены на рисунке 4.1, на котором представлена зависимость средней точности распознавания от количества коэффициентов.

Здесь и далее для обучения сети применялись чистые изолированные ноты базы данных музыкальных инструментов университета Айовы [31]. Для каждого инструмента были выбраны ноты в их основном диапазоне (2-3 октавы) длинной около 1-2 секунды.

В процессе обучения применялась валидационная выборка, составляющая 20% от всех данных. Для тестирования применялись ноты музыкальных инструментов, исполненные в другим стиле, например если обучение производилось на нотах сыгранных на *форте*, то тестирование производилось на данных сыгранных на *меццо-форте*. Такой подход приблизил результаты к тестированию в реальной среде. Объем тестовой выборки составил примерно 20% от обучающей выборки.

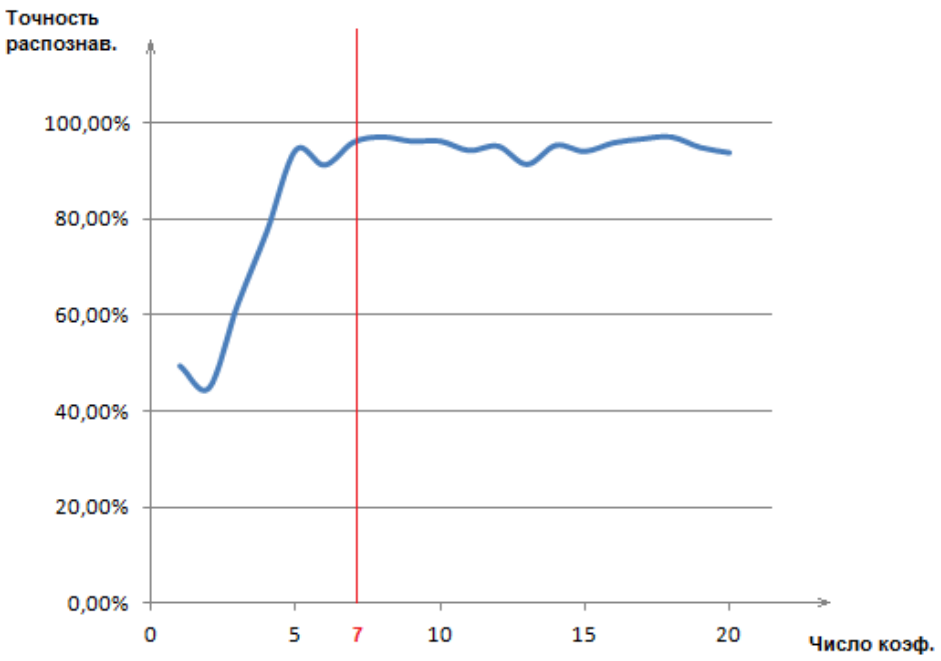

Рисунок 4.1. Зависимость точности распознавания от числа MFCC

Из представленного рисунка видно, что семь коэффициентов является достаточным для получения высокой точности распознавания.

Однако не только число признаков влияет на сложность решения, но и сложность пространства данных, для снижения сложности пространства данным был использован метод главных компонент (PCA).

4.2 Анализ эффективности метода главных компонент

Метод главных компонент (principle component analysis, PCA) позволяет снизить сложность пространства данных и соответственно позволяет найти более простое решение разделяющее данные на логические группы (кластеры). На рисунке 4.2 представлены данные трех первых MFCC компонентов, а на рисунке 4.3 представлены три главных компонента. Из рисунков видно, что кластеры на рисунке 4.3 имеют более компактную структуру, это можно оценить статистически. Компактность также подтверждается значениями размахов кластеров, приведенными в таблице 4.1. Размах (spread) – это величина, характеризующая степень компактности данных в кластере, чем она меньше, тем данные более компактны, а соответственно кластеры являются

лучше разделимыми. Данная величина вычисляется как сумма евклидового расстояния каждой точки кластера до центра кластера деленная на количество точек.

Улучшить характеристики кластеров стало возможным за счет устранения избыточности данных – устранения линейных комбинаций и повышения плотности данных.

Таблица 4.1 – Оценка размаха (spread) кластеров

Данные	1 кластер (труба)	2 кластер (скрипка)	3 кластер (рояль)
3 MFCC	0,18	0,19	0,24
3 PCA	0,16	0,15	0,19

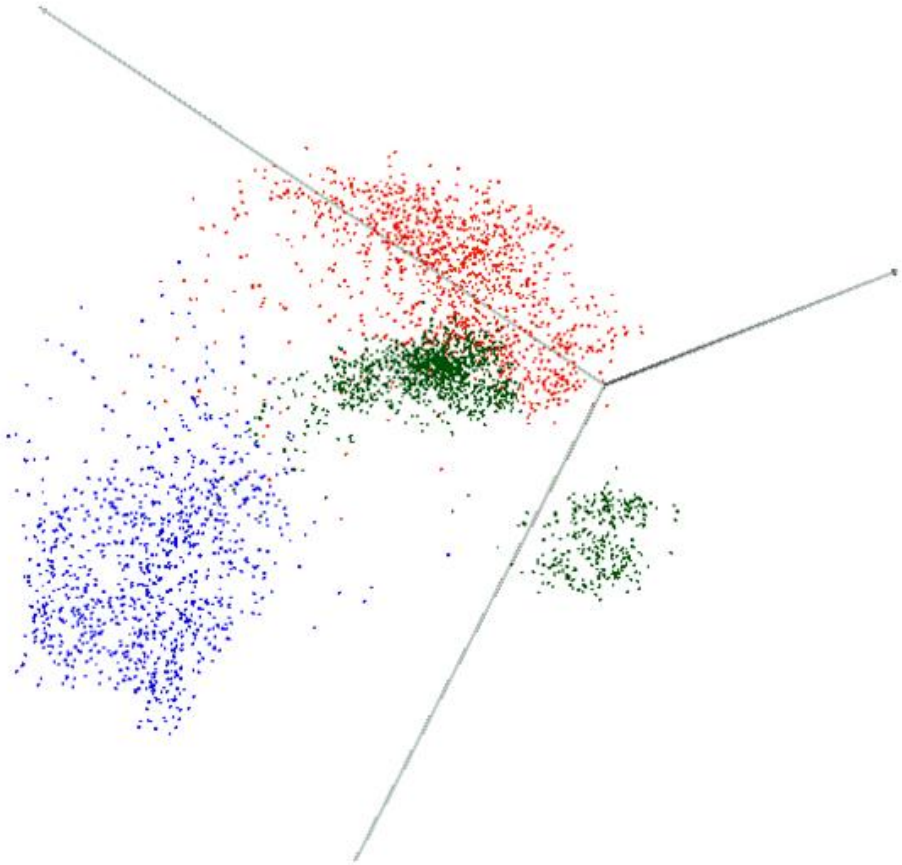

Рисунок 4.2 – Кластерная структура данных: три первых MFC коэффициента для трех инструментов

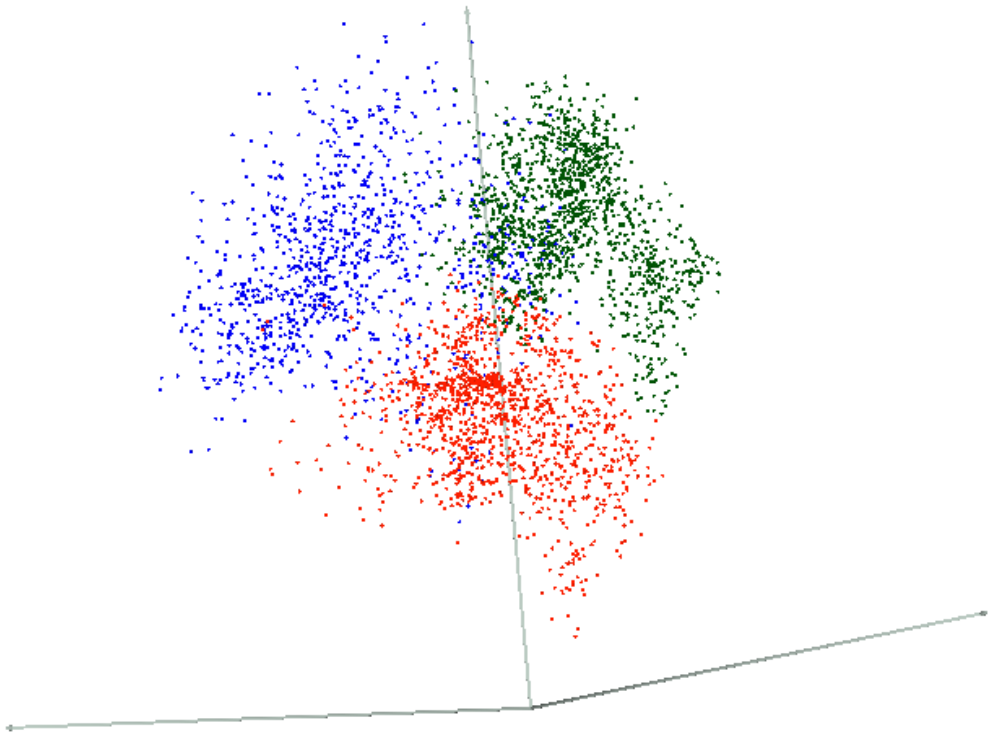

Рисунок 4.3 – Кластерная структура данных: три главных компонента для трех инструментов

Также лучшая разделимость кластеров подтверждается более быстрым обучением искусственной нейронной сети. В таблицы 4.2 представлена оценка среднего количества итераций затрачиваемых на обучение сети.

Таблица 4.2 – Оценка количества итераций обучения

Данные / Число инструментов	**2**	**3**	**5**	**6**
MFCC	41	71	104	187
PCA	5	26	53	75

Теперь рассмотрим вопрос выбора количества главных компонентов.

На рисунке 4.4 представлена зависимость точности распознавания от числа главных компонент для 16, 12 и 8 MFC коэффициентов соответственно. Из полученных данных можно заключить, для получения хорошей точности распознавания достаточно 6 главных компонентов, полученных из 8 мел-частотных кепстральных коэффициентов.

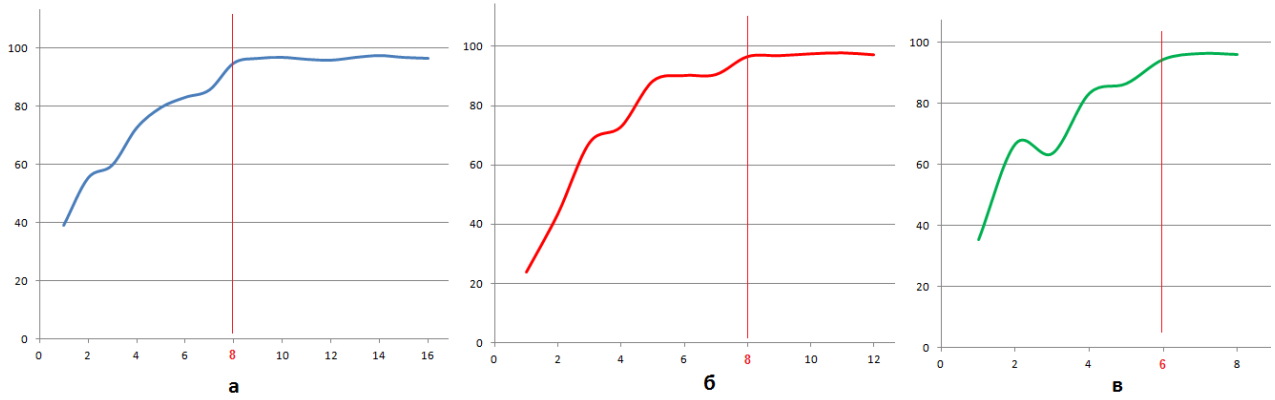

Рисунок 4.4 – Анализ числа главных компонент и точности распознавания (а – 20 MFCC, б – 12 MFCC, в – 8 MFCC)

4.3 Анализ применения алгоритма ESP

В данной работе был использован нейроэволюционный алгоритм ESP, который позволяет подобрать оптимальную структуру сети и произвести ее обучение. Однако данный алгоритм не показал лучшие результаты по точности распознавания, но потребовал значительно большего времени для обучения сети в сравнении с методом обратного распространения ошибки (BackPropagation). Сравнение ESP и BackPropogation приведено в таблице 4.3.

Таблица 4.3 – Сравнение алгоритмов обучения сети

Алгоритм	Кол-во инструментов	Время, мин	Точность, %
BackPropagation	2	0,1	99,10
ESP	2	10	98,35
BackPropagation	3	0,5	98,44
ESP	3	21	97,12
BackPropagation	4	1	97,72
ESP	4	30	95,10
BackPropagation	5	1,2	96,32
ESP	5	54	93,70

В связи с полученными данными было принято использовать алгоритм BackPropogation в качество основного, однако алгоритм ESP целесообразно использовать для подбора оптимальной структуры сети (количества нейронов).

4.4 Анализ точности распознавания

4.4.1 Анализ точности по инструментам

В таблице 4.1 представлено количество совпадений полученных по каждому инструменту, в строках – фактический инструмент, в столбцах – число совпадений по инструменту.

Таблица 4.1 – Результат тестирования (по инструментам)

	Труба	Скрипка	Рояль	Флейта	Кларнет	Тромбон	Маримба
Труба	114	-	-	-	-	-	-
Скрипка	-	37	1	-	-	4	2
Рояль	-	-	21	2	-	-	-
Флейта	-	3	62	2	-	-	-
Кларнет	-	1	1	4	57	-	2
Тромбон	-	-	2	-	-	257	10
Маримба	-	1	2	-	-	-	68

В таблице 4.2 представлена итоговая точность, полученная по каждому инструменту. Точность различна для каждого инструмента, наибольшую точность распознавания имеет труба, наименьшую – скрипка.

Таблица 4.2 – Точность распознавания (по инструментам)

№	**Инструмент**	**Точность, %**
1	Труба	100
2	Маримба	95,77
3	Тромбон	95,54
4	Флейта	92,65
5	Рояль	91,30
6	Кларнет	87,69
7	Скрипка	84,09
	Средняя	**94,34**

На следующем рисунке представлена зависимость точности распознавания от количества инструментов.

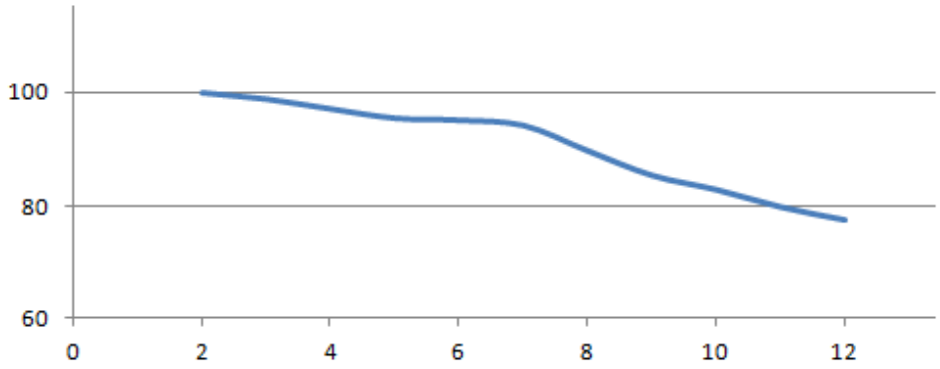

Рисунок 4.5 – Зависимость точности распознавания от количества инструментов

Алгоритм показывает хорошую точность, свыше 90 % для 7 и менее инструментов, однако при дальнейшем увеличении их числа наблюдается более значительный спад точности.

4.4.2 Сравнительный анализ с другими исследованиями

В таблице 4.3 представлен анализ точности распознавания в сравнении с другими исследованиями. Результаты исследований нельзя сравнивать напрямую, так как в них используются разные инструменты, разные музыкальные диапазоны и различное число признаков. Поэтому для наглядности сравнение производиться по количеству инструментов, участвующих в исследовании, как по наиболее общему признаку.

Таблица 4.3 — Сравнения точности распознавания с другими исследованиями

Автор	Число инструментов	Число признаков	Точность, %
Róisín Loughran, Jacqueline Walker, Michael O'Neill, Marion O'Farrell	3	4	95,88
Данное исследование		**6**	**98,9**
Aleksandr Diment, Padmanabhan Rajar, Toni Heitolla, Tuomas Virtanen	4	32	96
Данное исследование		**6**	**97,19**
Loisin Loughran, Jacquiline Walker, Michael O'Neill	5	20-95	60-64%
Данное исследование		**6**	**95,58**
Chandwadkar, Sutaone	6	160	90-92
Данное исследование		**6**	**95,22**
D. G. Bhalke, C. B. Rama Rao and D. S. Bormane	12	28	94,68
Rui Rui, Changchun Bao		85	87,5
Данное исследование		**6**	**77,85**

4.5 Результаты анализа сложного сигнала

Также используя данный алгоритм, удалось идентифицировать с хорошей точностью сольные партии музыкальных инструментов в полифоническом произведении. В качестве примера была взято результат музыкальное произведение композитора Шостаковича — Прелюдия №5. В произведении исполняют партии два музыкальных инструмента — фортепиано и скрипка. Партия фортепиано исполняется непрерывно, однако в данном произведении можно выделить 4 участка, два из которых — соло фортепиано, другие — соло скрипки. В результате работы алгоритма удалось распознать указанные сольные партии, результат работы программы представлен на рисунке 4.6 (вверху — амплитудно-временная зависимость сигнала, внизу — результат работы классификатора)

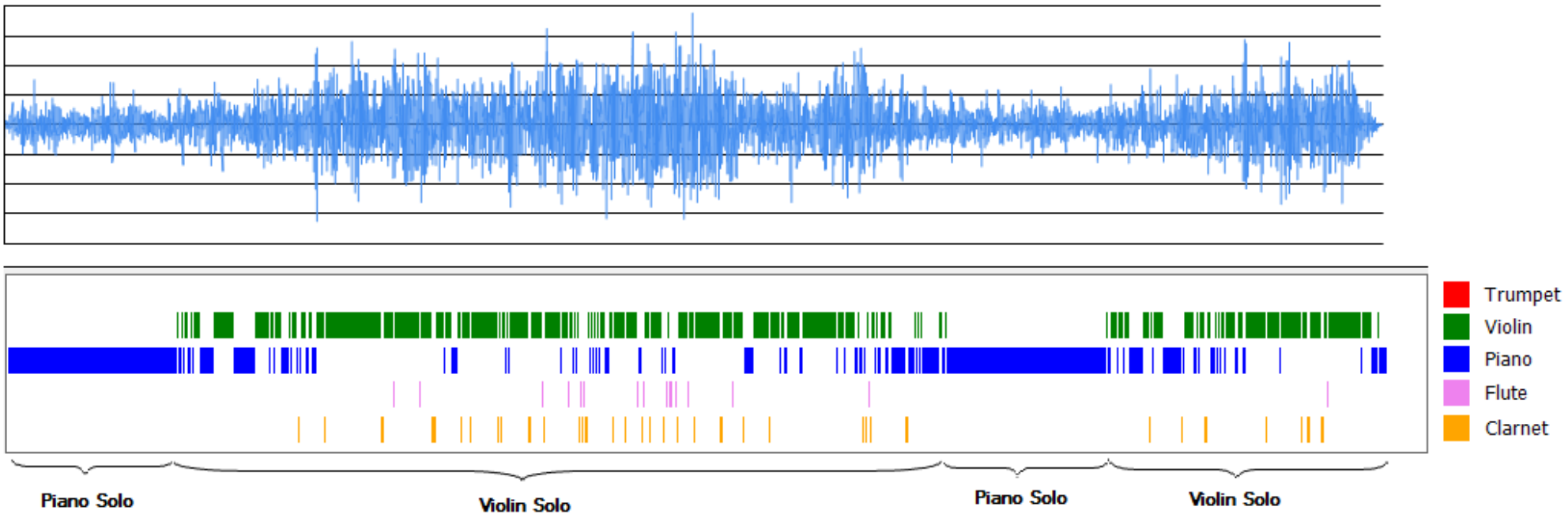

Рисунок 4.6 – Идентификация сольных партий в сложном музыкальном сигнале

ЗАКЛЮЧЕНИЕ

В ходе научно-исследовательской практики были решены следующие задачи:

1. Изучены и проанализированы характеристики звуковых музыкальных сигналов

2. Выбран значимый набор признаков для задачи распознавания, а именно мел-частотные кепстральные коэффициенты

3. Снижена сложность пространства признаков и найдено оптимальное их число за счет применения метода главных компонент.

3. Разработан алгоритм распознавания музыкальных инструментов с использованием искусственных нейронных сетей

4. Подготовлены обучающие и тестовые данные

5. Выполнена реализация алгоритма в виде программного пакета на языке C#.

Результаты данной работы могут быть использованы для решения задачи аннотирования медиаконтента для использования в поисковых системах и генерации плейлистов. Также разработанный алгоритм может рассматриваться как первый этап сегментации сложных музыкальных сигналов и идентификации музыкальных объектов (нот).

По итогам работы был опубликован ряд докладов в сборниках X Международной научно-практической конференции «Ключевые аспекты научной деятельности – 2014» (Польша), XX Международной научно-практической конференции «Современные техника и технологии» (г. Томск), I Международной научно-практической конференции «Информационные технологии в науке, управлении, социальной сфере и медицине» (г. Томск).

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Ф. В. Станкевич Использование мел-кепстральных коэффициентов для нейросетевого распознавания тембра музыкального инструмента. Сборник трудов конференции «Ключевые аспекты научной деятельности – 2014» Том 18, Математика. Физика. Современные информационные технологии. Пшемысль: Наука и образование, 2014 г.
2. Brown, J. C. (1999) Computer identification of musical instruments using pattern recognition with cepstral coefficients as features. *Acoust. Soc.*
3. Loisin Loughran, Jacquiline Walker, Michael O'Neill (2009). An Exploration of Genetic Algorithms for Efficient Musical Instrument Identification. *IET Irish Signals and Systems Conference.*
4. Eronen, A. (2011) Comparison of features for musical instrument recognition. *Workshop on Signal Processing for Audio and Acoustics (WASPAA)* 19-22.
5. Nielsen, A. B. Sigurdsson, S. Hansen, L. and Arenas-Garcia J. (2007) On the relevance of spectral features for instrument classification. *Proc. Acoustics, Speech and Signal Processing (ICASSP-07).*
6. Davis, S. Mermelstein, P. (1980). *Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences*. In IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. 28 No. 4, pp. 357-366.
7. Practical Cryptography, Mel Frequency Cepstral Coefficient (MFCC) tutorial, Access Date: 07.03.2014, URL: http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs
8. Хайкин С. Нейронные сети: полный курс, 2-е издание. Пер. с англ. — М.: Издательский дом "Вильямс", 2006. — 1104 с.
9. Faustino Gomez. Robust Non-Linear Control through Neuroevolution. PhD thesis, Department of Computer Sciences, 2003.
10. СанПиН 2.2.2/2.4.1340 – 03. Санитарно-эпидемиологические правила и нормативы «Гигиенические требования к персональным электронно-

вычислительным машинам и организации работы». – М.: Госкомсанэпиднадзор, 2003.

11. СНиП 23-05-95. Строительные нормы и правила "Естественное и искусственное освещение"
12. СНиП 21 – 01 – 97. Пожарная безопасность зданий и сооружений. М.: Гострой России, 1997. – с.12.
13. ГОСТ 17.2.1. 03-84. Охрана природы. Атмосфера. Термины и определения контроля загрязнения.
14. ГОСТ 17.4.3.04-85. Охрана природы. Почвы. Общие требования к контролю и охране от загрязнения.
15. ГОСТ 17.1.3.13-86. Охрана природы. Гидросфера. Общие требования к охране поверхностных вод от загрязнения.
16. ППБ 01–03. Правила пожарной безопасности в Российской Федерации. – М.: Министерство Российской Федерации по делам гражданской обороны, чрезвычайным ситуациям и ликвидации последствий стихийных бедствий, 2003.
17. Fadeev, A. (2008) Identification of Musical Object based on Continuous Wavelet Transformation. PhD thesis. Tomsk Polytechnic University.
18. Altman, V. (1990) *Hearing System*. Leningrad: “Science” publishing house.
19. Chandwadkar, Sutaone (2012) Role of Features and Classifiers on Accuracy of Identification of Musical Instruments. *2nd National Conference on Computational Intelligence and Signal Processing (CISP).*
20. Peeters, G. (2004) A large set of audio features for sound description (similarity and classification) in the CUIDADO project. *Proc. 115th AES Convention.*
21. American National Standards Institute, “American National psychoacoustical terminology” S3.20, 1973, American Standards Association.
22. Hollins, M; Bensmaia S.J. (2007). "The Coding of Roughness". *Canadian Journal of Experimental Psychology* 61 (3): 184–195.

23. Chandwadkar, Sutaone (2012) A Novel Supervised Learning Algorithm for Musical Instrument Classification. *International Conference on Telecommunications and Signal Processing.*
24. Perfecto Herrera, Xavier Amatriain, Eloi Balle, Xavier Serra (2000) Towards instrument segmentation for music content description: a critical review of instrument classification techniques. *International Conference on Music Information Retrieval.*
25. Vapnik, V. (1998) *Structural learning theory* New York: Willey.
26. Pawlak, Z. (1982) Rough sets. *Journal of Computer and Information Science* 11 (5) 341-356.
27. Dimet, A. Rajan, P. Heitolla, T. and Virtanen, T. (2013) Modified Group Delay Feature for Musical Instrument Recognition. *The 10th International Symposium on Computer Music Multidisciplinary Research.*
28. D. G. Bhalke, C. B. Rama Rao and D. S. Bormane (2014) Hybridization of Fractional Fourier Transform and Acoustic Features for Musical Instrument Recognition. *International Journal of Signal Processing, Image Processing and Pattern Recognition*
29. Sumit Kumar Banchhor, Arif Khan (2012) Musical Instrument Recognition using Zero Musical Instrument Recognition using Zero Crossing Rate and Short-time Energy. *International Journal of Applied Information Systems (IJAIS)*
30. Róisín Loughran, Jacqueline Walker, Michael O'Neill, Marion O'Farrell The Use of Mel-frequency Cepstral Coefficients in Musical Instrument Recognition. Ann Arbor, MI: MPublishing, University of Michigan Library
31. Musical Instruments Samples Access Date: 09.06.2014, URL: http://theremin.music.uiowa.edu/MIS.html

Приложение А

Раздел 1

METHODS & LITERATURE REVIEW

METHODS & LITERATURE REVIEW

1 Musical sound

Sound is a vibration that propagates as a mechanical wave of pressure and displacement, through some medium (such as air or water). Usually *sound* refers only to vibrations with frequencies within the range of hearing for humans.

Musical sound is a particular case of the sound. It is characterized by its pitch, timbre, loudness and duration. Pitch has range from note **A0** (sub-contra octave) to note **C8** (the five-line octave) (from 27.5Hz to 4186Hz) — a grand royal range. Sound timbre is defined by its overtones (harmonics) and depends on the source of the sound. Sound loudness must be less than the pain threshold and greater than the minimal sensitive threshold. Duration is determined by time interval from the beginning of the sound till the end of its dying. Sound of the most musical instruments can be illustrated on figure 1 as an amplitude-time dependency.

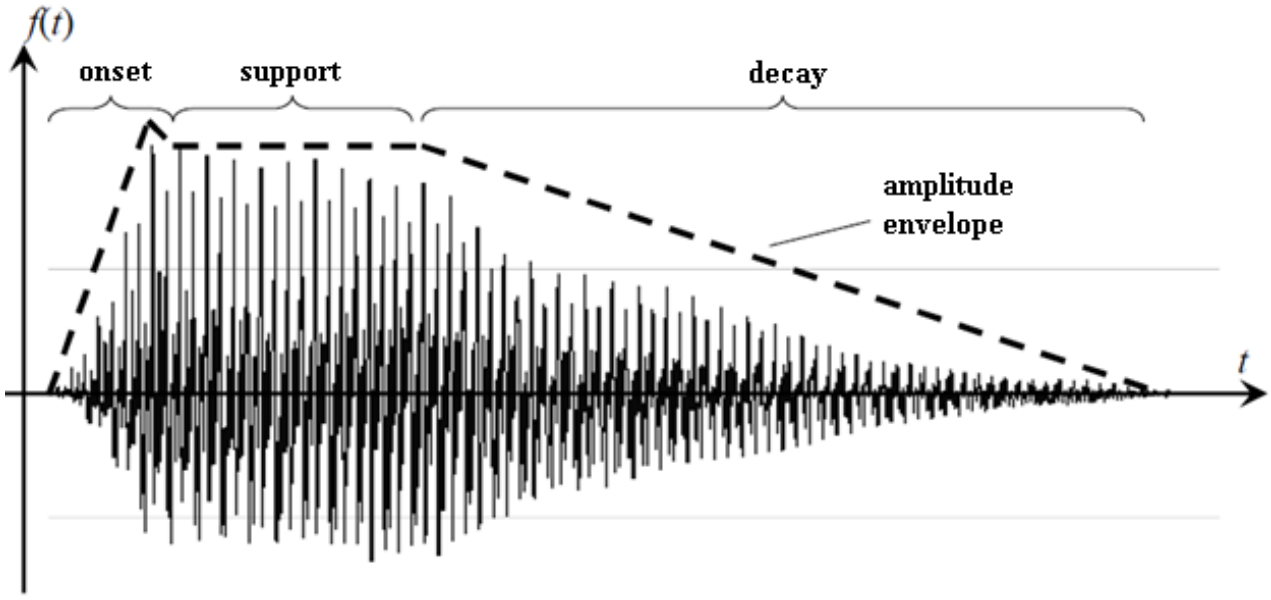

Figure 1 Amplitude-time dependency of a musical sound [17]

Onset, support and decay create an amplitude envelope. Typically a musical sound starts from fast rising of amplitude that is called onset. The duration of onset depends on the musical instrument and playing style and can vary from several to hundred and more milliseconds. During onset we have some additional components in spectrum because of wobbling of finger, hammer, valve etc. The onset is followed by support. The support is a stage with conditionally constant amplitude and

spectrum. The last stage of musical sound is the decay. The decay is characterized by a gradual decrease of the sound amplitude. The decay process is connected with termination of external force which produces the sound therefore the spectrum can differ from the support stage. The onset and support are the main stages which define the timbre. In its turn timbre allows drawing connection between a certain musical instrument and its sound.

Sound timbre also known as tone color, is the quality of a musical instrument sound or human voice. Timbre of the sound determines its tone and depends on the source of the sound, number of overtones associated with the fundamental tone and their amplitude [18]. The fundamental tone and overtones of musical sound are shown in frequency domain on figure 2.

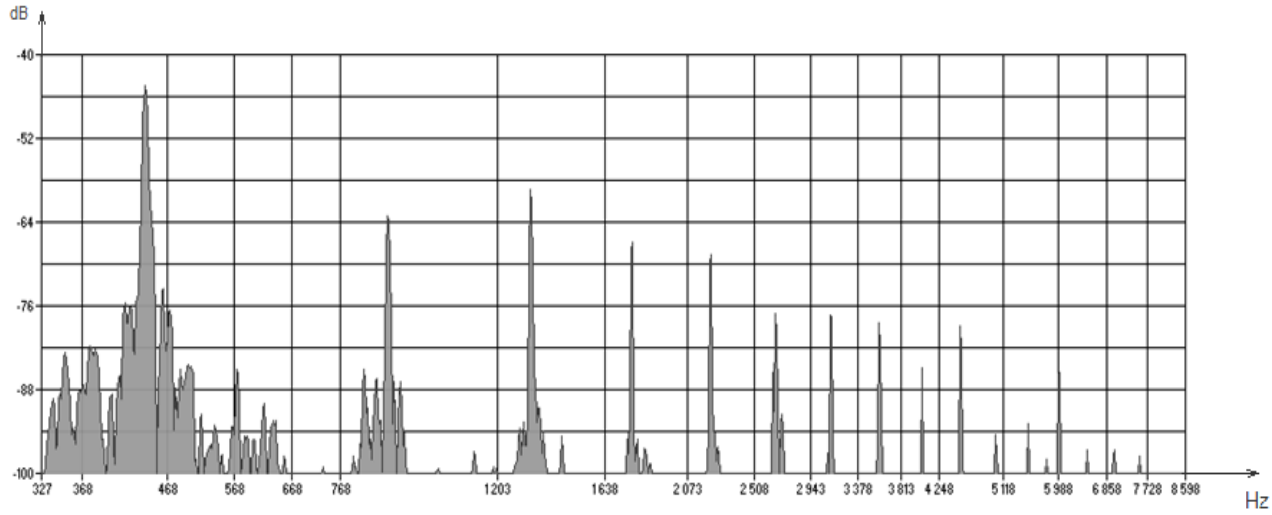

Figure 2 — Fundamental tone and overtones in frequency domain [17]

Sound timbre depends on musical instrument construction features and the way of sound extraction. Furthermore, a musician influences the sound timbre by changing the force and sharpness of the beginning or the ending of the sound and changing the loudness during the sound.

So we can imply that the musical instrument is defined by its timbre and the one musical instruments family should have similar timbre due to similar constriction. That means we can identify a certain musical instrument by analyzing its sound features such as timbre.

2 Feature extraction

The task of musical instrument identifying refers to classification tasks. The most feasible classification approach in this context is supervised learning i.e. training of classifier on labeled data. First and important thing in supervised learning is to find a proper feature space. So we are going to discuss feature extraction. In many studies lots of features were used for classification of musical instruments sound. In this section the review of the main of them is provided.

Basically, we can classify sound features as:

- temporal
- spectral
- perceptual

Let us try to consider each of them.

Temporal features are features that are extracted in temporal domain. List of the most popular temporal features is shown below.

- *Rise time* is the time taken by a signal to change from a specified low value to a specified high value.
- *Decay time* is the time taken by a signal to change from a specified high value to a specified low value.
- *Temporal centriod* is the point in time in a signal where most of the energy is located on average.
- *Zero-crossing rate* (ZCR) is an indicator for noisiness of the signal which is often used in speech recognition. ZCR is calculated by counting the number of times that signal cross zero in time domain within a given window.
- *Autocorrelation coefficients* are measures of how well signal matches with a time shifted version of itself [19].

Spectral features are features which extract in spectral domain. List of the most popular spectral features is given below.

- *Spectral envelope* is the curve that describes an envelope of the spectrum, i.e. it wraps tightly around the magnitude spectrum, linking the peaks.

- *Spectral centroid* is a measure used in digital signal processing to characterize a spectrum. It indicates where the mass center of the spectrum is. Perceptually, it has an explicit connection with impression of brightness of the sound [20]. It is calculated as the weighted mean of the frequencies presented in the signal which are determined using Fourier transform with their magnitudes as the weights.
- *Spectral spread* is a measure of variance of the spectrum around the mean value.
- *Spectral skewness* is a measure of the asymmetry of the distribution around the man value. The skewness is calculated from 3^{rd} order moment.
- *Spectral kurtosis* indicates the flatness or peakedness of the energy distribution. Higher kurtosis means more of the variance is the result of infrequent extreme deviations, as opposed to frequent modestly sized deviations. It is calculated from the 4^{th} order moment.
- *Spectral slope* is a measure of how quickly the spectrum of an audio sound tails off towards to the high frequencies. It is calculated using a linear regression. The spectral slope gives an indication of the rate of decrease of the magnitude of the spectrum.
- *Spectral roll-off* is another measure of spectral shape. It is the point where frequency that is below some percentage (usually 95%) of the spectrum resides. It is often used as an indicator of the skew of the frequencies percent in a window [20].

The last features are perceptual features. It is a special type of features which are based on human perception of the sound. Mainly used of them are listed below.

- Loudness is the characteristic of a sound that is primarily a psychological correlate of physical strength (amplitude). More formally, it is defined as "that attribute of auditory sensation in terms of which sounds can be ordered on a scale extending from quiet to loud" [21].

- *Roughness* is simply how rough it sounds. Roughness is studied by examining how textures are perceived and encoded by an individual's somatosensory system [22].
- *Sharpness* is a measure of the high frequency content of a sound, the greater the proportion of high frequencies the ‘sharper’ the sound.
- *Mel-Frequency Cepstral Coefficients* (MFCC) are generally used features. They are used as standard features in speech recognition. The MFCC calculation algorithm can be described in the following steps:

    1. Take the Fourier transform (windowed) of a signal.
    2. Map the powers of the spectrum onto the mel-scale, using triangular overlapping windows.
    3. Take the logs of the powers at each of the mel-frequencies.
    4. Take the discrete cosine transform of mel log powers.
    5. The MFCCs are the amplitudes of the resulting spectrum [20].

Some of researchers tried to use single features e.g. MFCC [2] and most of them used combination of these features to get better result.

3 Classification

Classification of gained features may be performed by various classifiers. In this section we are going to consider the most popular classifiers which are used for recognition of musical instruments.

The *K-Nearest Neighbors* algorithm is one of the most popular algorithms for instance based learning. It first stores features of all training data and then finds a set of *k* nearest training examples for classifying a new instance then refer this instance to the class which has more examples in the set. It is quite easy to implement this algorithm, but there are several drawbacks: it does not provide generalization mechanism, it requires all training data in memory, and it is highly sensitive to irrelevant features [23].

The *Naive Bayesian Classifier* is based on estimation of probabilities of each class and the conditional probabilities of each feature within their frequencies over the training data. The set of these estimations corresponds to the learned hypothesis

which is formed by counting the frequency of various data combinations within the training examples and this can be used to classify a new instance [24]. This method is "naive" because it suggests a certain distribution function across the features e.g. Gaussian distribution that can be incorrect. However, in practice this method can be useful within non-complicated dependencies in feature space.

Discriminant Analysis is a technique that is related to multivariate analysis of variance and multiple regression. Discriminant analysis tried to minimize the ration within-class scatter to the between-class scatter and builds a definite decision region between classes. It provides linear, quadratic or logistic function of variables that better separate classes into predefined groups. One possible drawback is its reduced generalization power [24].

Binary tree is constricted top-down, beginning with the feature that seems to be the most informative one, that is, the one that maximally reduces entropy. Branches are then created from each one of the different values of this descriptor. The training examples are sorted the appropriate descendant node, and this process in then repeated recursively using the examples of one of the descendant nodes, then with others. Once the tree has been built, it can be pruned to avoid overfitting and to remove secondary features. Although building a binary three is a recursive procedure, it is faster than the training a neural network [24].

Support Vector Machine (SVM) is a recently discovered method which is based on statistical learning theory [25]. The main principle of SVM is to search the optimal linear hyperplane within minimization of expected classification error for unseen test examples. According to the structural risk minimization principle, the function classifies the training data accurately and which belongs to a set of functions with the lowest complexity will generalize best regardless of higher input space dimension. The SVM, based on this principle, uses a systematic approach to find a leaner function with the lowest complexity. For linearly non-separable data, SVM can map nonlinearly input to higher dimensional feature space to find a linear hyperplane in new space. Although, there is no guaranty that a linear solution exists in a high-

dimensional feature space, but in practice it is feasible to construct a working solution [24].

Artificial Neural Network is a computational model inspired by animals' central nervous systems (in particular the brain). It is usually presented as a system of interconnected "neurons" that can compute values from inputs by feeding information through the network. Depending on structure and neuron activation functions there are several types of ANNs such as multilayered perceptron, time-delay network, radial basis function network, self-organizing maps and etc. Due to similarity to brain ANN has a good capability for generalization.

Rough Sets [26] are a novel technique for evaluating the relevance of the features used for description and classification. In rough set theory any set of similar or indiscernible objects is called an elementary set and forms a basic granule of knowledge. On the other hand, the set of discernible objects are considered rough (vague). Vague concepts cannot be described in terms of their elements information, but they may be replaced by two precise concepts, called lower and approximation respectively. Low approximation contains all objects that surely belong to the concepts where the upper approximation includes all objects that possiblly belong to the concept. This assignment is made thorough a probabilistic membership function. Then this information can be used for classifying purposes [24].

4 Related works review

Within this research more than 30 works for last 5 years (and some basic works which were published earlier) were reviewed. The most significant of them are described below.

In [19] is considered an approach based on combination of temporal, spectral, perceptual features and their statistical derivatives such as minimum and maximum value, average value, standard deviation and variance. Altogether they used 32 features which produce with their statistical values 160 (32*5) features. Recondition accuracy was about 90% for 6 musical instruments.

Loughran and others [3] used a genetic algorithm to minimize the number of features. They considered 95 various features. Due to genetic algorithm they

managed to decrease number of features to 20. Final accuracy was 60-64% for 5 musical instruments.

In [23] is used an algorithm based on non-negative matrix factorization (NMF). The dimension of feature space was 60. Finally, accuracy of 87% for 12 instruments was gained.

In paper [27] is proposed to use an additional set of features *Modified Group Delay Features* (16 features) with MFCC (16 features). Then the total number of features was 32. The gained accuracy was 96% for 4 musical instruments.

In [28] was proposed a hybrid method based on Fractional Fourier transformation that is calculated both on spectral and time domains. Overall 28 features (Fractal MFCC + temporal features) were used. The recognition accuracy was 94.68% for 12 instruments.

In [29] an analysis of two features (Zero-crossing rate, short-time energy) within recognition of three musical instruments was performed. The clear difference in these features for each instrument was shown. However, no average recognition accuracy was found in this article.

One interesting result was shown in [30]. MFCC was used with Principle Component Analysis (PCA) to reduce the dimension of feature space. This approach was tested on database of three instruments. The best accuracy was gained with 15 MFCC and four principle components of them.

The aggregated information of these studies is shown in table 1.

Table 1 — The analysis of related studies

Author	Year	Number of instruments	Number of features	Accuracy
1. Loisin Loughran, Jacquiline Walker, Michael O'Neill	2009	5	20-95	60-64%
2. Rui Rui, Changchun Bao	2012	12	85	87,5%
3. Chandwadkar, Sutaone	2012	6	160	90-92%
4. D. G. Bhalke, C. B. Rama Rao and D. S. Bormane	2014	12	28	94.68%
5. Róisín Loughran, Jacqueline Walker, Michael O'Neill, Marion O'Farrell	2008	3	4	95.88%
6. Aleksandr Diment, Padmanabhan Rajar, Toni Heitolla, Tuomas Virtanen	2013	4	32	96%
7. Sumit Kumar Banchhor, Arif Khan	2012	3	2	n/a

5 Problem statement & conclusion

Based on reviewed methods we can conclude that each of them has one or more limitation such as

- low recognition accuracy
- great number of features
- low performance
- small musical intervals
- decreasing accuracy with rising number of instruments

So in this work we aim to develop an algorithm with good accuracy and performance within minimization of features number. To implement this task we should perform subtasks such as

- Feature selection
- Test data preparing
- Building of classifier
- Training of classifier
- Algorithm optimization
- Feature space minimization

Based on reviewed studies (particular on [2], [3], [4] and [5]) Mel-frequency cepstral coefficients appear as the best features for musical instrument recognition, that is why it seems to be feasible to use MFCCs (or some of them) as a main set of features and find some other features (temporal or spectral) to get better recognition accuracy. Furthermore, this approach can allow minimizing the number of used features at all.

For better results, first of all, the parameters of MFCC calculation algorithm should be optimized, and then additional features should be considered.