Веб-приложение для распознавания музыкальных инструментов и преобразования аудиозаписей

Авторы:
Бухараев Алим
Прохоров Юрий
Савелов Михаил
Яушев Фарух

Руководитель:
Бабичев Сергей Леонидович

2019

Содержание

1 Техническое задание

1.1 Аннотация

Данный проект — курсовая работа вышеперечисленных студентов второго курса ФУПМ (факультета управления и прикладной математики) МФТИ. Он состоит из 3 основных обособленных частей:

1. Библиотека и API на C++ для обработки аудиозаписей.
2. API в виде расширения PHP для обработки аудиозаписей.
3. Пользовательский интерфейс в виде веб-приложения.

Об установке можно прочитать ниже. Ссылка на репозиторию Github:

```
https://github.com/suchusername/sounds
```

Изначально, у нас была идея написать программу, способную распознавать музыкальные инструменты на аудиозаписи. Причина связана с тем, что эта задача не имеет точных результатов, и мы хотели получить свое собственное частное решение. Затем мы решили расширить функционал нашего приложения, добавив наиболее популярные запросы, и реализовать его как легко доступный пользователю веб-сервис. Таким образом, основные цели проекта:

- Повышение командных навыков.
- Реализация приложения по популярной тематике.
- Применение знаний языка C++.
- Изучение новых инструментов и технологий.

Пользователю предлагается загрузить свой файл в формате WAV на сервер. После этого он получает некоторую статистику о файле и выбирает, какое действие он желает выполнить с ним или сразу с несколькими файлами. Среди возможных функций: предсказание музыкального инструмента по аудиозаписи, слияние файлов, их обрезка, усиление громкости, ускорение и другие преобразования. Аудиозапись хорошего качества может быть использована далее для дальнейшего обучения нашей или каких-либо других моделей. Наконец, преобразованный файл пользователь может прослушать и загрузить обратно на свое устройство.

При разработке приложения использовались следующие технологии:

- Язык C++ стандарта 11 для написания серверной составляющей (ядра).
- Набор инструментов для написания расширения PHP на C++.
- Python 3 и библиотеки Keras, NumPy, SciPy для реализации нейронной сети.
- Средства PHP, HTML5, CSS и Javascript для разработки пользовательского интерфейса.

1.2 Цели

Главной мотивацией работы является его выполнение в качестве курсового проекта по информатике студентов второго курса ФУПМ МФТИ: Бухараева Алима, Прохорова Юрия, Савелова Михаила и Яушева Фаруха. Руководитель проекта — Бабичев Сергей Леонидович.

Другая важная цель — реализация классификатора мелодий по играющему музыкальному инструменту. Нами был проведен анализ уже существующих средств распознавания музыкальных инструментов, и, как оказалось, тема не является очень популярной. Мы же считаем, что данный этап предобработки аудиозаписей принесет большую пользу для дальнейшего их анализа (например, распознавания играющей композиции, удаления некоторых тональностей или инструментов), поэтому мы попробовали внести свой вклад в решение данной задачи.

Еще одна задача проекта — это разработка доступного пользовательского интерфейса для простейшей обработки аудиофайлов. Мы не нашли сайта, который представляет собой "онлайн-редактор аудиозаписей", и в связи с этим решили дополнить нашу программу этими функциями и предоставить ее как веб-сервис.

Наконец, важной для нас целью является обретение опыта работы в команде и грамотное распределение обязанностей. Кроме того, сюда же входит получение опыта приложения полученных знаний о языке C++ к реальной задаче, изучение новых полезных для практического применения технологий и инструментов.

1.3 Требования

В данном разделе перечислены все задачи и подзадачи, которые мы поставили перед собой в рамках выполнения настоящей проектной работы.

1. Библиотека и API на C++.
    - Разработка объектно-ориентированной внутренней структуры библиотеки.
    - Побайтовый разбор файла формата WAV.
    - Реализация функционала
        - Вывод статистики о файле.
        - Изменение громкости.
        - Ускорение аудиозаписи.
        - Слияние аудиозаписей.
        - Предсказание музыкального инструмента.
        - Обрезка аудизаписи.
    - Универсальный пользовательский интерфейс с вышеперечисленными функциями.
2. Нейронная сеть.
    - Подготовка и нарезка датасета для обучения модели.
    - Поиск оптимального способа предобработки аудиозаписей.
    - Моделирование наиболее эффективной архитектуры классификатора, различающего музыкальные инструменты.
    - Обучение классификатора на подготовленных данных.
    - Анализ результатов.
3. Расширение PHP.
    - Настройка Apache веб-сервера и установка необходимых инструментов для реализации PHP-расширения.
    - Универсальный пользовательский интерфейс с вышеперечисленными функциями.
4. Веб-приложение.
    - Загрузка нескольких файлов одновременно.
    - Выбор файла/файлов, которые требуется преобразовать.
    - Просмотр статистики об аудиозаписи.
    - Загрузка файла с сервера обратно.

1.4 Распределение обязанностей

Член команды	Задача
Бухараев Алим	Изучение сверточных и рекуррентных нейронных сетей
	Поиск большого числа мелодий, пригодных для использования в качестве обучающей выборки
	Проектирование и обучение модели-классификатора
	Подготовка презентации
Прохоров Юрий	Реализация анализатора файла формата .WAV
	Различные преобразования аудиозаписей и реализация библиотеки C++
	Реализация программного интерфейса расширения PHP
	Написание отчета и документации
Савелов Михаил	Изучение PHP, HTML5, JavaScript и CSS
	Моделирование пользовательского интерфейса
	Разработка структуры веб-страницы
	Дизайн
Яушев Фарух	Изучение PHP, HTML5, JavaScript и CSS
	Моделирование пользовательского интерфейса
	Разработка структуры веб-страницы
	Дизайн

2 Документация

2.1 Структура директорий в репозитории

В этом и следующем разделах приведено описание репозитории проекта на GitHub.

Audios

В данную директорию загружаются все пользовательские файлы, там же создаются новые промежуточные аудиофайлы.

Classes

В этой директории находятся реализации методов всех классов на C++. Каждому классу соответствует свой файл. Классы, отвечающие преобразованиям аудиозаписей вынесены в отдельную папку *Classes/Queries*.

Documentation

В этой директории находится данный документ и его .tex файл. Также в папке *Documentation/Papers* присутствуют статьи на разные тематики, которые использовались при написании этой проектной работы.

Instrument_classifier

В этой директории находятся следующие файлы:

- *launch.py*
  Содержит реализацию нейронной сети с использованием библиотеки Keras. Этот же файл отвечает за получения предсказания на конкретной аудиозаписи. Нейронную сеть можно явно использовать следующим образом:

  ```
  cd Instrument_classifier
  python3 launch.py path/to/file.wav
  ```

- *my_model_*.h5*
  Различные версии модели.

root

Данная директория является корневой директорией Apache веб-сервера. В ней содержатся следующие объекты:

- *.php*
  Файлы формата .php — ответы сервера на запросы клиента. Они генерируют html код, который видит пользователь.
- *sounds_update.sh*
  Shell-скрипт, который делает новую сборку расширения PHP, переустанавливает его и перезапускает Apache веб-сервер.
- *logs*
  Текстовый файл с отладочной информацией.
- *root/css, root/js*
  Эти папки содержат файлы соответствующего формата, необходимые для реализации дизайна и пользовательского интерфейса веб-страницы.
- *root/sounds*
  Эта папка содержит все файлы, необходимые для создания расширения PHP. Среди них:
  - *config.m4*
    Файл с параметрами для компилиции расширения PHP.

– *php_sounds.h*
Задание интерфейса PHP-расширения.
– *sounds.cc*
Реализация методов PHP-расширения.

2.2 Конкретные файлы в репозитории

config.h

В этом файле содержатся все константы, используемые во всех файлах.

server.h

В этом файле содержится объявление всех классов и их методов. Также там присутствует описания функций программного интерфейса.

sounds_cpp_api.cc

Реализация методов программного интерфейса.

main.cc

Этот файл предназначен только для тестирования различных функций.

2.3 Документация: библиотека C++

В этом разделе перечислен весь функционал библиотеки с указаниями, как им пользоваться. Также к некоторым методам приведены примеры использования.

Информация об аудиозаписи

```
vector<double> sounds_info(const string &name);
```

`name` — полный путь к файлу относительно текущей директории.

Возвращает вектор из 6 элементов:

`[0]` — Размер файла в байтах.

`[1]` — Число аудиоканалов (1 — моно, 2 — стерео).

`[2]` — Sample rate (количество значений амплитуды каждую секунду), обычно 44100.

`[3]` — Bit depth (число битов на каждый sample).

`[4]` — Количество sample'ов в аудиофайле.

`[5]` — Продолжительность аудиофайла (в секундах).

Если элемент вектора `[0]` $= -1$, то ошибка.

Распознавание музыкальных инструментов

```
vector<double> sounds_classify(const string &name);
```

`name` — полный путь к файлу относительно текущей директории.

Возвращает вектор из 5 элементов с вероятностями музыкальных инструментов:

`[0]` — аккордион

`[1]` — пианино

`[2]` — скрипка

`[3]` — гитара

`[4]` — флейта

Если элемент вектора `[0]` $= -1$, то ошибка.

Обрезка аудиозаписей

```
string sounds_crop(const string &name, const string &new_name,
        double left = 0, double right = -1);
```

`name` — полный путь к файлу относительно текущей директории
`new_name` — полный путь с новым названием преобразованного файла
`left = 0` — левая граница обрезки в миллисекундах (default: начало файла)
`right = -1` — правая граница обрезки в миллисекундах (default: конец файла)

Если `left` ≤ 0 или $\geq$ длины файла, то левая граница — начало файла. Если `right` ≤ 0 или $\geq$ длины файла, то правая граница — конец файла. По умолчанию, границы — начало и конец файла соответственно.

Обрезает аудиозапись и сохраняет ее в указанную директорию под выбранным пользователем именем.

Возвращает строку:

- `"OK"` — успешно
- `string` — строка с текстом ошибки, если неуспешно

Изменение громкости

```
string sounds_volume(const string &name, const string &new_name,
        double k, double left = 0, double right = -1, bool smooth = false);
```

`name` — полный путь к файлу относительно текущей директории
`new_name` — полный путь с новым названием преобразованного файла
`k` — множитель громкости (0 — тишина, 1 — без изменений)
`left = 0` — левая граница в миллисекундах (default: начало файла)
`right = -1` — правая граница в миллисекундах (default: конец файла)
`smooth = false` — плавное изменение громкости (линейное, в течение 2 секунд)

Если `left` ≤ 0 или $\geq$ длины файла, то левая граница — начало файла. Если `right` ≤ 0 или $\geq$ длины файла, то правая граница — конец файла. По умолчанию, границы — начало и конец файла соответственно.

Если `k` < 0, то ошибка, если `k` $= 0$, то тишина, если $0 <$ `k` < 1, то аудиозапись становится тише, если `k` > 1, то аудиозапись становится громче. **Внимание:** если выбрать `k` очень большим (обычно ~ 5), то для нормального звучания не хватает битовой глубины, и звук становится обрывистым (что иногда звучит забавно).

Изменяет громкость выбранного отрезка аудиозаписи и сохраняет ее в указанную директорию под выбранным пользователем именем.

Возвращает строку:

- `"OK"` — успешно
- `string` — строка с текстом ошибки, если неуспешно

Изменение скорости аудиозаписи

```
string sounds_speed(const string &name, const string &new_name,
        double mult);
```

`name` — полный путь к файлу относительно текущей директории
`new_name` — полный путь с новым названием преобразованного файла
`mult` — множитель скорость аудиозаписи (1 — без изменений)

Если `mult` = 0, то ошибка. Если 0 < `mult` < 1, то аудиозапись становится медленнее в `1/mult` раз. Если `mult` = 1, то без изменений. Если `mult` > 1, то аудиозапись ускоряется в `mult` раз. Если `mult` < 0, то аудиозапись проигрывается в обратную сторону (инвертируется).

Изменяет скорость аудиозаписи и сохраняет ее в указанную директорию под выбранным пользователем именем.

Возвращает строку:

- `"OK"` — успешно
- `string` — строка с текстом ошибки, если неуспешно

Слияние аудиозаписей

```
string sounds_merge(const string &left, const string &right,
        const string &new_name);
```

`left` — полный путь к первому файлу относительно текущей директории
`right` — полный путь ко второму файлу
`new_name` — полный путь с новым названием преобразованного файла

Сливает две аудиозаписи в одну и сохраняет ее в указанную директорию под выбранным пользователем именем.

Возвращает строку:

- `"OK"` — успешно
- `string` — строка с текстом ошибки, если неуспешно

2.4 Документация: расширение PHP

В этом разделе перечислен весь функционал расширения PHP с указаниями, как им пользоваться. Также к некоторым методам приведены примеры использования. Здесь написано то же самое, что и в разделе по библиотеку C++, но в терминах PHP.

Информация об аудиозаписи

```
sounds_info($name);
```

`name` — полный путь к файлу относительно текущей директории.

Возвращает массив из 6 элементов:

`[0]` — Размер файла в байтах.

`[1]` — Число аудиоканалов (1 — моно, 2 — стерео).

`[2]` — Sample rate (количество значений амплитуды каждую секунду), обычно 44100.

`[3]` — Bit depth (число битов на каждый sample).

`[4]` — Количество sample'ов в аудиофайле.

`[5]` — Продолжительность аудиофайла (в секундах).

Если элемент массива `[0]` = −1, то ошибка.

Распознавание музыкальных инструментов

```
sounds_classify($name);
```

`name` — полный путь к файлу относительно текущей директории.

Возвращает массив из 5 элементов с вероятностями музыкальных инструментов:

`[0]` — аккордион

`[1]` — пианино

`[2]` — скрипка

`[3]` — гитара

`[4]` — флейта

Если элемент массива `[0]` $= -1$, то ошибка.

Обрезка аудиозаписей

```
sounds_crop($name, $new_name, $left = 0, $right = -1);
```

`name` — полный путь к файлу относительно текущей директории
`new_name` — полный путь с новым названием преобразованного файла
`left = 0` — левая граница обрезки в миллисекундах (default: начало файла)
`right = -1` — правая граница обрезки в миллисекундах (default: конец файла)

Если `left` ≤ 0 или $\geq$ длины файла, то левая граница — начало файла. Если `right` ≤ 0 или $\geq$ длины файла, то правая граница — конец файла. По умолчанию, границы — начало и конец файла соответственно.

Обрезает аудиозапись и сохраняет ее в указанную директорию под выбранным пользователем именем.

Возвращает строку:

- `"OK"` — успешно
- `string` — строка с текстом ошибки, если неуспешно

Изменение громкости

```
sounds_volume($name, $new_name, $k, $left = 0, $right = -1, $smooth = False);
```

`name` — полный путь к файлу относительно текущей директории
`new_name` — полный путь с новым названием преобразованного файла
`k` — множитель громкости (0 — тишина, 1 — без изменений)
`left = 0` — левая граница в миллисекундах (default: начало файла)
`right = -1` — правая граница в миллисекундах (default: конец файла)
`smooth = False` — плавное изменение громкости (линейное, в течение 2 секунд)

Если `left` ≤ 0 или $\geq$ длины файла, то левая граница — начало файла. Если `right` ≤ 0 или $\geq$ длины файла, то правая граница — конец файла. По умолчанию, границы — начало и конец файла соответственно.

Если `k` < 0, то ошибка, если `k` $= 0$, то тишина, если $0 <$ `k` < 1, то аудиозапись становится тише, если `k` > 1, то аудиозапись становится громче. **Внимание:** если выбрать `k` очень большим (обычно ~ 5), то для нормального звучания не хватает битовой глубины, и звук становится обрывистым (что иногда звучит забавно).

Изменяет громкость выбранного отрезка аудиозаписи и сохраняет ее в указанную директорию под выбранным пользователем именем.

Возвращает строку:

- `"OK"` — успешно

- `string` — строка с текстом ошибки, если неуспешно

Изменение скорости аудиозаписи

```
sounds_speed($name, $new_name, $mult);
```

`name` — полный путь к файлу относительно текущей директории
`new_name` — полный путь с новым названием преобразованного файла
`mult` — множитель скорость аудиозаписи (1 — без изменений)

Если `mult` = 0, то ошибка. Если 0 < `mult` < 1, то аудиозапись становится медленнее в `1/mult` раз. Если `mult` = 1, то без изменений. Если `mult` > 1, то аудиозапись ускоряется в `mult` раз. Если `mult` < 0, то аудиозапись проигрывается в обратную сторону (инвертируется).

Изменяет скорость аудиозаписи и сохраняет ее в указанную директорию под выбранным пользователем именем.

Возвращает строку:

- `"OK"` — успешно
- `string` — строка с текстом ошибки, если неуспешно

Слияние аудиозаписей

```
sounds_merge($left, $right, $new_name);
```

`left` — полный путь к первому файлу относительно текущей директории
`right` — полный путь ко второму файлу
`new_name` — полный путь с новым названием преобразованного файла

Сливает две аудиозаписи в одну и сохраняет ее в указанную директорию под выбранным пользователем именем.

Возвращает строку:

- `"OK"` — успешно
- `string` — строка с текстом ошибки, если неуспешно

3 Результаты

3.1 Веб-приложение

Пользовательский интерфейс реализован в виде клиент-серверного приложения, которое доступно через браузер.

При открытии страницы пользователь сразу видит рабочее поле приложения. Он имеет возможность загрузить один или несколько файлов формата WAV. В дальнейшем будет возможно загружать и обрабатывать файлы и других форматов. Загруженные файлы отображаются в центральном поле. Далее клиент может выбрать один из загруженных на сервер файлов и применить к нему одну из следующих операций:

1. Просмотреть статистику о файле:
   - частота дискретизации, число аудиоканалов, длительность и прочее
   - спектрограмма
2. Воспроизвести файл в браузере.
3. Преобразовать аудиозапись:
   - увеличить громкость
   - ускорить (в частности, воспроизвести в обратную сторону)
   - обрезать
   - слить с другой аудиозаписью
   - переименовать
4. Распознать музыкальный инструмент на аудиозаписи:
   - аккордион
   - пианино
   - скрипка
   - гитара
   - флейта
5. Скачать загруженную или преобразованную аудиозапись.

3.2 Нейронная сеть

Для работы с аудиофайлами обычно используются рекуррентные нейронные сети. Тем не менее задачу классификации аудиозаписей мы свели к задаче классификации изображений и использовали для этого более простой тип нейронных сетей — свёрточные.

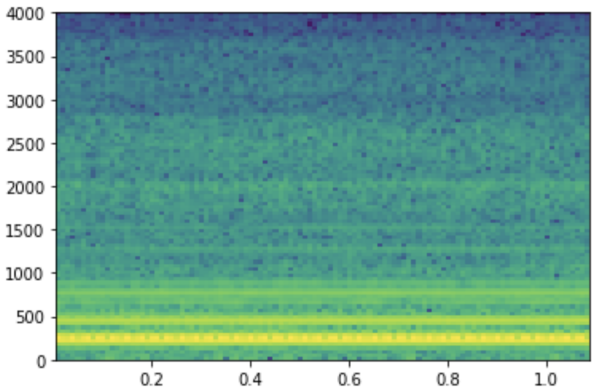

Рис. 1: Спектрограмма одной секунды звучания флейты. Разрешение 101 x 549.

Исходный аудиофайл разбивался на фрагменты длиной в одну секунду, после чего для каждого из них строилась спектрограмма. Все такие спектрограммы подавались на вход нейронной сети, которая по картинке определяла, что за инструмент звучит в данном фрагменте. Архитектура модели была тщательно подобрана, что подтверждается высоким процентом корректных ответов на тестовых данных.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 101, 549, 1)	0
dropout_1 (Dropout)	(None, 101, 549, 1)	0
padding_0 (ZeroPadding2D)	(None, 103, 551, 1)	0
conv_0 (Conv2D)	(None, 101, 138, 5)	50
activation_1 (Activation)	(None, 101, 138, 5)	0
max_pool_0 (MaxPooling2D)	(None, 50, 69, 5)	0
padding_1 (ZeroPadding2D)	(None, 52, 71, 5)	0
conv_1 (Conv2D)	(None, 50, 69, 20)	920
activation_2 (Activation)	(None, 50, 69, 20)	0
max_pool_1 (MaxPooling2D)	(None, 25, 34, 20)	0
padding_2 (ZeroPadding2D)	(None, 29, 38, 20)	0
conv_2 (Conv2D)	(None, 13, 17, 40)	20040
activation_3 (Activation)	(None, 13, 17, 40)	0
flatten_1 (Flatten)	(None, 8840)	0
dropout_2 (Dropout)	(None, 8840)	0
hidden (Dense)	(None, 20)	176820
fc (Dense)	(None, 5)	105

Total params: 197,935
Trainable params: 197,935
Non-trainable params: 0

Рис. 2: Архитектура модели.

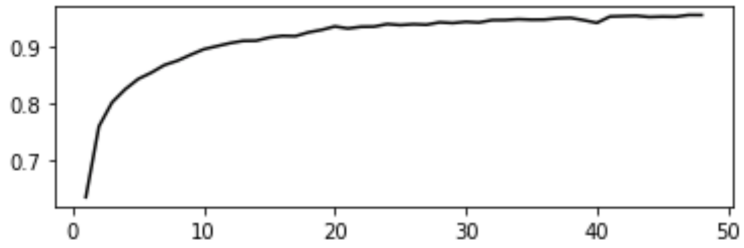

Рис. 3: Процесс обучения модели. 50 эпох. Достигается точность в 95%.

Обучение проводилось на 150 аудиозаписях, скаченных с YouTube. Для пяти инструментов: аккордиона, пианино, скрипки, гитары и флейты - было скачено по 30 файлов, на каждом из которых инструмент звучал без сопровождения.

Наиболее успешной показала себя модель, архитектура и график обучения которой приведены выше. Модели тестировались на 50 произвольных аудиофайлах (по 10 на каждый инструмент), также взятых с YouTube.

Как мы видим, на двух записях фортепьяно модель сделала неправильные предсказания. Объяснить это можно тем, что на первой из них у инструмента между молоточками и струнами были вставлены

специальные металлические пластинки, делающие звук более гитарным, а на второй представлен некий далёко не идеальный VST-плагин.

В целом, результаты обучения можно считать успешными, поскольку были получены рабочий датасет, способ препроцессинга и архитектура модели, а значит, эксперимент можно продолжить, используя большее количество инструментов.

guitar	piano	violin	accordion	flute
YAMAHA F310	HonkyTonk Piano	Yamaha V3 Student Violin Outfit, 4/4	Scandalli Air 1	Pearl 795E Elegante Flute
Fender CF-60	Yamaha P45	Student 1/16 Violin by Gear4music	Paolo Soprani Studio 72	Pearl 505E Beginner Flute
Fender PM-1	Yamaha C7	Stentor Student II Violin Outfit 4/4	Scandalli model Intense	JFL301E
Martin DX1 RAE	Fazioli F278	Yamaha V7SG Intermediate Violin, 4/4 Size	Vignoni Ravel III Accordion	16 Flutes
Takamine GD10 NS	KAWAI K500	Yamaha V10SG Intermediate Violin Pack, 4/4 Size	Hohner Accordion	Yamaha Wood Flute (YFL894W)
DOWINA D555	Steinway O	Maurizio Tadioli	Soprani Professionale 1A	Yamaha Flute YFL-481
TAKAMINE G320	EZkeys vst Acoustic Grand Piano	Daniela Solca	Saban tuye version	Yamaha 261
Fender FA-115	Yamaha Garritan Abbey Road Studios	Lorenzo Storioni	Excalibur Accordion Geneva Ultralite 24 Bass	Yamaha 221
Fender FA-125	QUANTUM LEAP PIANOS Steinway	Hidersine Veracini	Weltmeister Perle 48 Bass	Yamaha 371
TANGLEWOOD TFA XB	AvantGrand NU1X Bösendorfer Imperial Sound	Stentor Conservatoire Violin Outfit 4/4	Brandoni 72 Bass Compact 65W1 Mahogany	YAMAHA 481-H

Рис. 4: Результат работы модели. Красным выделены неправильно угаданные аудиозаписи.

3.3 Внутренняя реализация

После загрузки аудиозаписей на сервер, там происходит побайтовая ”распаковка” файла (рис. 6). Это позволяет точно определить необходимые для последующей обработки параметры файла.

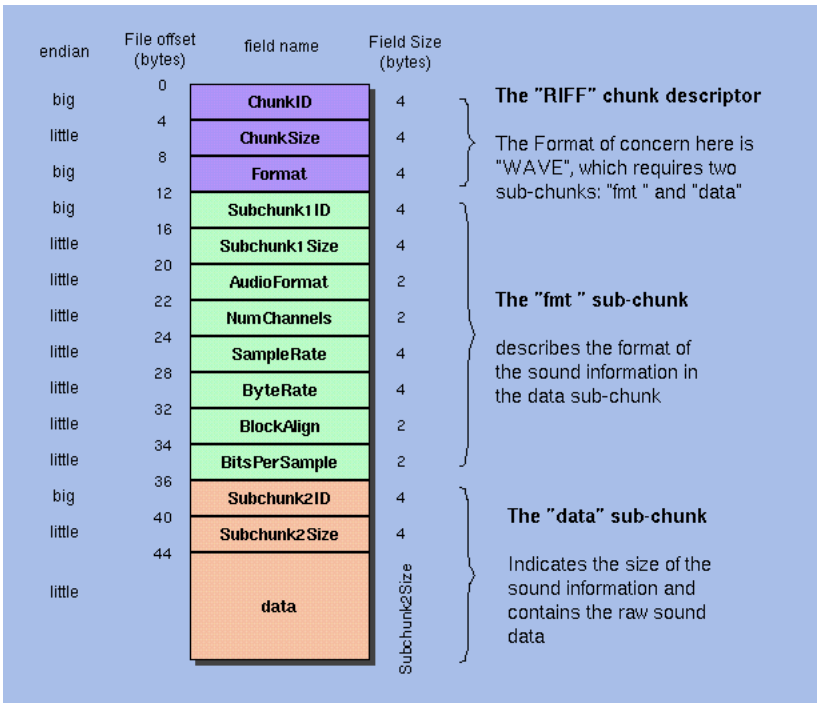

Рис. 5: Внутренняя структура файла формата WAV.

Когда пользователь выбирает функцию, которая подразумевает непосредственную работу с содержимым аудиофайла (например, классификация по музыкальному инструменту или обрезка), то вызывается соответствующая функция расширения PHP. Расширение PHP представляет собой модуль с определенным набором методов, который загружается наравне со встроенными модулями языка PHP. Модуль вызывает метод из скомпилированной заранее библиотеки C++ (использует объектный .so файл). Схематически вызовы изображены на рисунке 6.

Если выбранная операция является преобразованием аудиофайла, то соответствующий библиотечный метод выполняет это операцию и сохраняет новый файл под другим именем. Если пользователь запросил классификацию аудиозаписи, то на сервере запускается интерпретатор Python3, который получает пред-

сказание заранее обученной нейронной сети.

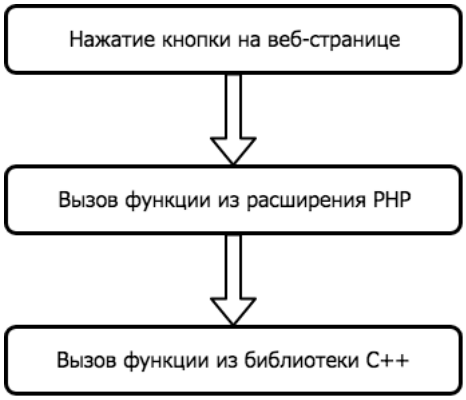

Рис. 6: Схема вызова методов.

Работа веб-приложения возможна в параллельном режиме, что осуществляется веб-сервером Apache. Для каждой пользовательской сессии извлекается ее уникальный 128-битный идентификатор сессии, и для каждого такого подключения создается директория, чтобы избегать конфликтов между различными пользователями. Если сессия неактивна в течение определенного времени, то директория сессии на сервере удаляется.

4 Источники

1. Hojjat Salehinejad, Sharan Sankar: Recent Advances in Recurrent Neural Networks, 2017.
   URL: https://arxiv.org/abs/1801.01078

2. Asifullah Khan, Anabia Sohail: A Survey of the Recent Architectures of Deep Convolutional Neural Networks, 2019.
   URL: https://arxiv.org/abs/1901.06032

3. Mingqing Yun, Jing Bi: Deep Learning for Musical Instrument Recognition, 2017

4. Apache HTTP Server Version 2.4 Documentation
   URL: https://httpd.apache.org/docs/2.4/

5. WAVE PCM soundfile format
   URL: http://soundfile.sapp.org/doc/WaveFormat/

6. Sara Goleman: Extension Writing Part I: Introduction to PHP and Zend, 2005
   URL: https://devzone.zend.com/303/extension-writing-part-i-introduction-to-php-and-zend/

7. Sara Goleman: Extension Writing Part II: Parameters, Arrays, and ZVALs, 2005
   URL: https://devzone.zend.com/317/extension-writing-part-ii-parameters-arrays-and-zvals/
   URL: https://devzone.zend.com/318/extension-writing-part-ii-parameters-arrays-and-zvals-continued/