**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

|  |  |  |
| --- | --- | --- |
| СОГЛАСОВАНО  Научный руководитель,  доцент департамента  программной инженерии факультета компьютерных наук,  канд. техн. наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Р.З.Ахметсафина  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2017 г. |  | УТВЕРЖДАЮ  Академический руководитель образовательной программы «Программная инженерия»  профессор департамента программной инженерии, канд. техн. наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Шилов  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2017 г. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** |  | | **Программа для адаптации музыкального произведения под определенный стиль на основе машинного обучения**  **Пояснительная записка**  **ЛИСТ УТВЕРЖДЕНИЯ**  **RU.17701729.темы\_нет\_в\_лмс ТЗ 01-1-ЛУ** | | |
|  |  | |
| Исполнитель  студент группы БПИ 175  \_\_\_\_\_\_\_\_\_\_\_\_ / И. О. Балбин /  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2017 г. | |
|  | | |
|  | |  |

**Москва 2017**

|  |  |  |  |
| --- | --- | --- | --- |
| УТВЕРЖДЕН  RU.17701729.\_\_\_ ТЗ 01-1-ЛУ | |  |  |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** |  | | **Программа для адаптации музыкального произведения под определенный стиль на основе машинного обучения**  **Техническое задание**  **RU.17701729.темы\_нет\_в\_лмс ТЗ 01-1**  **Листов 18** | | |
|  | |  |
|  |
| **Москва 2017** | | |

**СОДЕРЖАНИЕ**

[1. ВВЕДЕНИЕ 4](#_Toc483132025)

[1.1. Наименование программы 4](#_Toc483132026)

[1.2. Краткая характеристика области применения 4](#_Toc483132027)

[2. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ 5](#_Toc483132028)

[2.1. Функциональное назнаечние 5](#_Toc483132029)

[2.2. Эксплутационное назначение 5](#_Toc483132030)

[3. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ 6](#_Toc483132031)

[3.1. Постановка задачи 6](#_Toc483132032)

[3.2. Описание алгоритма и функционала программы 14](#_Toc483132033)

[3.3 Описание и обоснование выбора состава технических и программных средств 15](#_Toc483132034)

[3.3.1. Состав технических и программных средств 15](#_Toc483132035)

[3.3.2. Обоснование технических и пограммных средств 15](#_Toc483132036)

[4. ОЖИДАЕМЫЕ ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ 16](#_Toc483132037)

[4.1. Предпологаемая потребность 16](#_Toc483132038)

[4.2. Экономические преимущества разработки по сравнению с отечественными и  
зарубежными образцами или аналогами 16](#_Toc483132039)

[5. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 17](#_Toc483132040)

# 1. ВВЕДЕНИЕ

**1.1.** **Наименование** **программы**

Наименование программы – «Программа для адаптации музыкального произведения под определенный стиль на основе машинного обучени» (Programm for music composition adaptation to certain music style based on machine learnin).

**1.2.** **Краткая** **характеристика** **области** **применения**

Программа для адаптации музыкального произведения может служить инструментом для Аранжировщиков и Композиторов. С ее помощью можно получить прототип музыкального произведения на основе имеющихся композиций, искать идеи для своих музыкальных композиций и создавать полноценные работы, предварительно дообработав до желаемого результата полученную композицию.

Так же программа может служить в развлекательных целях.

# 2. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ

**2.1** **Функциональное** **назначение**

Функциональным назначением программы является создание адаптированного под определенный музыкальный стиль музыкального трека с помощью машинного обучения и нейронных сетей.

Если обрабатывается трек формата \*.wav, то адаптация идет под трек, который передает в качестве “стиля”. Если же обрабатывается трек формата \*.mid, то адаптация идет под стили из предложенного перечня стилей.

**2.2** **Эксплуатационное назначение**

Программа может быть использована на компьютерах, телефона, планшетах и т.д с браузером и доступом в интернет для создания новых музыкальных треков, адаптации их под новый стиль а так же в развлекательных целях.

# 3. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

**3.1.** **Постановка задачи**

* Программа должна предоставлять пользователю выбор формата загружаемого/ых треков
* Программа должна передавать треки для обработки на сервер
* После принятие сервером треков/а программа должна обработать с помощью нейросети
* После завершения обработки нейросети сервер должна возвращать результат (обработанный трек)

**3.2.** **Описание алгоритма и функционирования программы**

**3.2.1** **Алгоритм адаптации музыкальной дорожки**

Задача адаптации музыкального трека делится на две подзадачи, а именно стилизация \*.wav и стилизация \*.mid файлов.

**3.2.1.1** **Алгоритм адаптации для \*.wav файлов**

\*.wav файлы представляют собой аудиопоток, обладающий несколькими характеристиками (громкость, частота дискретизации…). Основная идея передачи стиля для \*.wav файлов заключается адаптированние neural-style (данный модель позволяет передавать стиль для картинок). Для этого \*.wav дорожки преобразовываем в спектрограммы, которые уже можем прогнать через neural-style. И полученную картинку преобразовываем обратно в \*.wav файл.

**3.2.1.1.1 Дизайн архитектуры (\*.wav)**

Baseline архитектуры – CNN (Convolution neural network). Сходящиеся нейросети позволяют обрабатывать данные с учетом взаимосвязей между ними (Обработка графов, пикселей картинок с учетом их взаимоположения). Эта архитектура позволяет высчитывать низкоуровневые признаки изображений.

В neural – style CNN модифицировано для передачи “стиля” изображения. Дизайн neural-style остался без изменений.

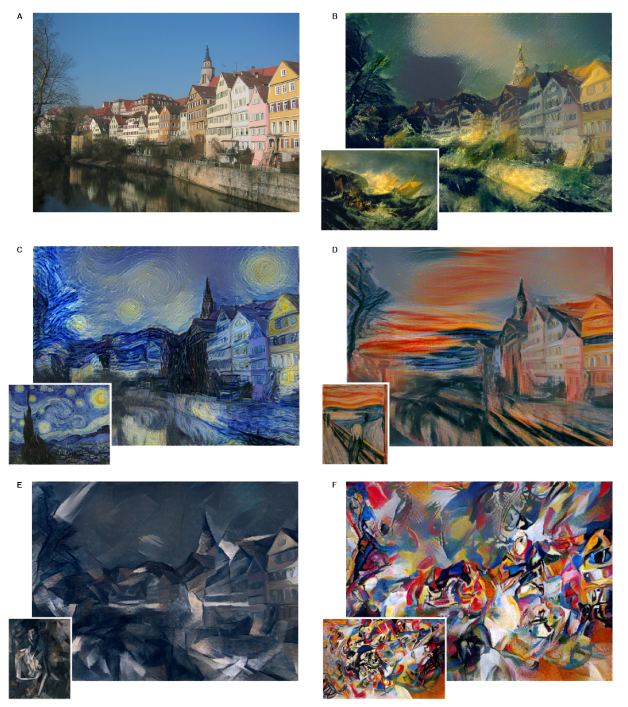


Рисунок 1. Примеры работы neural-style.

**3.2.1.1.2** **Preprocessing, Postprocessing (\*.wav)**

\*.wav файлы преобразовываем в STFT спектрограммы (STFT спектрограмма отображает Громкость(Dz) от времени и частоты).

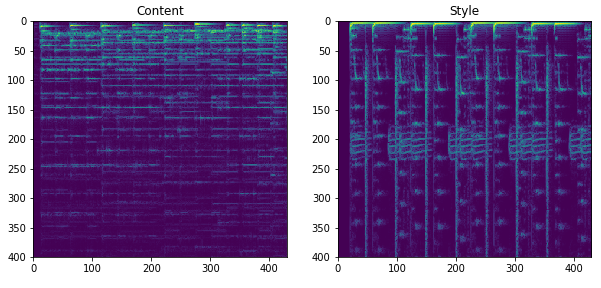


Рисунок 2. Спектрограммы wav файлов.

Для преобразования спектрограммы обратна в \*.wav файлов используем алгоритм Грииф-Лима.

**3.2.1.2** **Алгоритм адаптации для \*.mid файлов**

Формат \*.mid представляет собой набор команд, каждая из которых характеризуется собой несколькими показателями (Тип команды, канал, событие). Сам же формат mid позволяет хранить непосредственно нотные партии, а не просто музыкальную дорожку как частотно описанный звук.

**3.2.1.2.1 Дизайн архитектуры (\*.mid)**

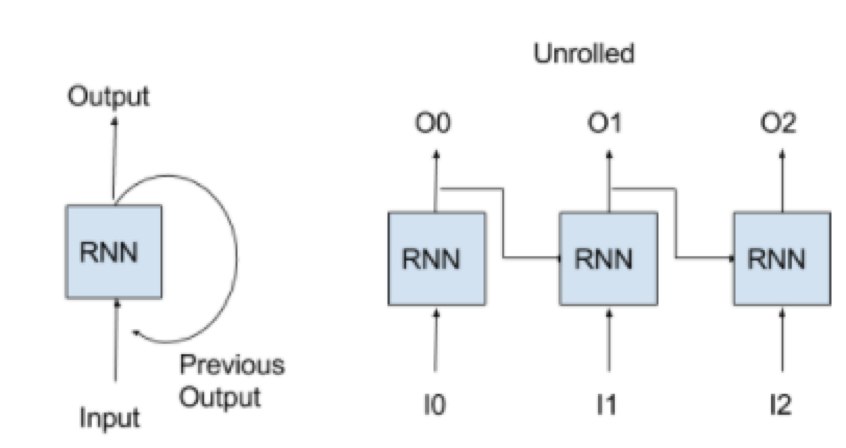
Задача адаптации музыкального произведения попадает под кластер задача seq2seq (последовательность в последовательность). Для решения подобного рода задач не подходят “прямые” нейросети т.к. их выход не зависит от прошлых результатов, что не позволяет обучить нейросеть строить связанные последовательности. Для решения подобного рода задач используют рекуррентные нейронные сети, которые получают на вход помимо выходов предыдущего слоя свой прошлый выход (рис.3). Такая архитектура позволяет генерировать “связанные” последовательности.

Рисунок 3. Архитектура простейшей рекуррентной нейронной сети – SimpeRNN.

(рекуррентный блок можно представить в виде связанной цепочки из одинаковых слоев)

(рекуррентный блок можно представить в виде связанной цепочки из одинаковых слоев)

# 

# Но SimpleRNN не способна улавливать долгосрочные связи. Для решения этой проблемы используются ее модификации. В частности с этим справляется LSTM (Long short-term memory) за счет модифицированного рекурентного блока.

# 

Рисунок 4. Архитектура LSTM

Ключевой компонент LSTM – это cell state (1) – горизонтальная линия, проходящая по верхней части схемы. Каждый блок LSTM состоит из четырех слоев, взаимодействующих особым образом. Первый слой SIGMA (2) возвращает значение в диапазоне от 0 до 1 и поэлементно умножается на выход ht-1, выступая в качестве “фильтра” для информации из прошлых выходов. Следующая пара слоев SIGMA и TANH (2) отвечают за добавление новой информации в нашу ячейку. И последний слой SIGMA (3) выступают в качестве фильтра.

Данная архитектура хорошо показывает себя в задач схожей нашей (Генерация текстов под определенный стиль, синтез речи и тд). Поэтому выбрана эта архитектура в качестве baseline.

**3.2.1.2.2 Preprocessing (\*.mid)**

Для кодирования использовалось One-hot кодирование. На каждый момент времени создается вектор из 1 и 0, где 1 – зажатая клавиша, 0 – не зажатая клавиша.

encoder = LabelBinarizer()

encoder.fit(notes)

joblib.dump(encoder, "encoders/LabelBinarizer.sav")

data = encoder.transform(notes)

Отсюда следуют следующая проблема – музыка непрерывна и нам нужно ее как – то дискретизировать. Для решения этой проблемы был выбран фиксированный интервал – одна четвериь. И кодирование файла происходило в интервалы кратных выбранному интервалу.

def get\_msg(file):

notes = []

n = 0

try:

midi = converter.parse(file)

print(file)

notes\_to\_parse = None

parts = instrument.partitionByInstrument(midi)

if parts: # file has instrument parts

notes\_to\_parse = parts.parts[0].recurse()

else: # file has notes in a flat structure

notes\_to\_parse = midi.flat.notes

prev\_offset = 0

for element in notes\_to\_parse:

new\_offset = 0.5

if prev\_offset == element.offset:

new\_offset = 0

else:

new\_offset = element.offset - prev\_offset

if 0 < new\_offset <= 0.5:

new\_offset = 0.5

if 0.5 < new\_offset <= 1:

new\_offset = 1

if 1 < new\_offset <= 1.5:

new\_offset = 1.5

if 1.5 < new\_offset:

new\_offset = 2

if isinstance(element, note.Note):

notes.append(str(element.pitch) + "|" + str(new\_offset) + "|" + str(element.octave))

n += 1

elif isinstance(element, chord.Chord):

notes.append('.'.join(str(n) for n in element.normalOrder) + "|" + str(new\_offset)

+ "|")

n += 1

prev\_offset = element.offset

except Exception:

print("Что - то не так")

return notes

После кодирования нашу дорожку нужно разбить на датасет (Последовательность нот + следующая за последовательностью нота). Для этого брались последовательностью длиной 2, и с шагом 1 разбивалась вся дорожка.

def create\_dataset(dataset, look\_back=1):

"""

Создает последовательность вида:

X = [n-look\_back, n-look\_back+1, ...., n-1] и Y = [n]

"""

dataX, dataY = [], []

for i in range(len(dataset) - look\_back - 1):

a = dataset[i:(i + look\_back)]

dataX.append(a)

dataY.append(dataset[i + look\_back])

return numpy.array(dataX), numpy.array(dataY)

Следующая проблема заключается в том, что ноты/аккорды в треке для адаптации могли не встречаться в музыкальных дорожках на которых мы изучали, в следствии чего мы не сможем прогнать через модель дорожку с “уникальными” аккордами/нотами (По сути сеть решает задачу классификации, стараясь предсказать на каждом шаге следующею ноту/аккорд ориентируясь на предыдущие. И если мы встречаем новую ноту, т.е. новый класс мы не можем обработать данный трек нашей сетью). Данную проблему можно решить составив словарь из всех нот в нашей обучающей выборки и с помощью ngram найти самый “похожий”.

notes\_set = set(notes)

ngram\_notes = list(notes\_set)

G = ngram.NGram(ngram\_notes)

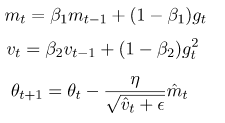
joblib.dump(G, "app/encoders/ngram.sav")

**3.2.1.2.3 Обучение (\*.mid)**

Обучение происходило на датасетах взятых из разных источников, каждый их которых представлял из себя выборку ~из 300 \*.mid треков выбранных по стилю.

Архитектура сети изображена на рис.2.

Функция ошибки: categorical cross entropy H(p,q) = - \sum_x p(x) \log(q(x))

Оптимизатор: Adam

Градиент считался с помощью алгоритма обратного распространения по minibatch размером 32.

model.add(LSTM(64, input\_shape=(look\_back, len(data[0]))))

model.add(Dense(32, activation='relu'))

model.add(Dense(len(data[0]), activation='softmax'))

model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['acc'])

**3.2.3 Сервер интерфейс и веб приложения.**

Сервер написан на фреймворке FLASK. В папке templates хранятся HTML шаблоны страниц. Модуль routes отвечает за логику работы сервера.

При запуске приложения открывается следующая страница:

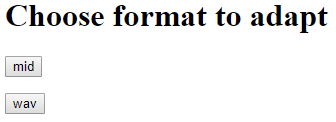


Рисунок 5. Начальный экран

* Кнопка “MID” – открыть страницу для загрузки и обработки \*.mid файла
* Кнопка “WAV” – открыть страницу для загрузки и обработки \*.wav файлов

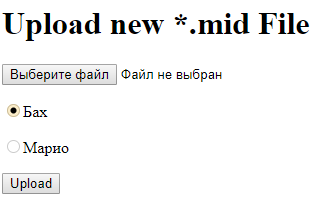


Рисунок 6. Экран для обработки \*.mid файла

кнопка

* Кнопка “Выберите файл” – открывает диалоговое окно для загрузки файла
* RadioButton [Бах, Марио] – выбор стиля
* Кнопка “Upload“ – загрузить файл и начать обработку (Если файл не загружен, то перенаправляет на эту же страницу)



Рисунок 6. Экран для обработки \*.wav файлов

* Кнопка ”Music” – открывает диалоговое окно для загрузки \*.wav файла, который будет адаптироваться
* Кнопка “Style” – открывает диалоговое окно для загрузки \*.wav файла, подс стиль которого будут адартироваться
* Кнопка “Upload” – загрузить файл и начать обработку (Если файлы не загружены, то перенаправляет на эту же страницу)



Рисунок 7. Экран загрузки адаптированного файла

* Кнопка “Download” – скачивает адаптированный трек.

**3.3.** **Описание и обоснование выбора состава технических и программных средств**

**3.3.1** **Состав технических и программных средств**

Для работы программы необходим следующий набор программных средств:

1. Браузер

Для работы программы необходим следующий набор технических средств:

1. Доступ в интернет

**3.3.2 Обоснование выбора состава технических и программных средств**

Интерфейс программы реализован в виде веб-приложения, для которого необходим браузер и веб-приложениие. Для стабильного работы которого требуется стабильный интернет и браузер.

# 4. ОЖИДАЕМЫЕ ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ

**4.1.** **Предполагаемая потребность**

Программа может быть использована в качестве вспомогательного инструмента для Аранжировки музыкальных композиций, с ее помощью Аранжировщики смогут получить полноценный прототип, предварительно доработав композицию.   
 Для Композиторов «MusicAdapt» может служить средством для поиска новых музыкальных нахождений, поиском идей для будущих работ.

Так же «MusicAdapt» может служить в развлекательных целях.

**4.2.** **Экономические преимущества разработки по сравнению с отечественными и  
зарубежными образцами или аналогами**

В рамках данной курсовой работы расчет экономических показателей не предполагается.

# 5. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Neural-style - https://github.com/jcjohnson/neural-style
2. ГОСТ 19.101-77 Виды программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001. – 126 с
3. ГОСТ 19.102-77 Стадии разработки. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001. – 126 с
4. ГОСТ 19.103-77 Обозначения программ и программных документов. //Единая система  
   программной документации. – М.: ИПК Издательство стандартов, 2001. – 126 с
5. ГОСТ 19.104-78 Основные надписи. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001. – 126 с
6. ГОСТ 19.105-78 Общие требования к программным документам. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001. – 126 с
7. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001. – 126 с
8. ГОСТ 19.404-79 Пояснительная записка. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001. – 126 с
9. ГОСТ 19.603-78 Общие правила внесения изменений. //Единая система программной  
   документации. – М.: ИПК Издательство стандартов, 2001. – 126 с
10. ГОСТ 19.604-78 Правила внесения изменений в программные документы, выполненные  
    печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001. – 126 с

# Приложение 1

## **Описание и функциональное назначение модулей**

Таблица 1 – Описание и функциональное назначение классов

|  |  |
| --- | --- |
| Модуль | Назначение |
| Wild\_card | Модуль, отвечающий за составление разбиение данных на датасет. |
| train | Модуль, отвечающий за составление и обучение нейросети для \*.mid. |
| Music controller | Модуль, отвечающий за парсинг и восстановления \*.mid файлов. |
| TF\_AFAPT | Моудль, отвечающий за адаптацию \*.wav файлов. |
| remix | Модуль, отвечающий за применение к \*.mid заранеее обученной модели. |
| routes | Модуль, отвечающий за логику работы сервера |
| сonfig | Настроечный файл сервера |
| \_\_init\_\_ | Модуль, выполняющиеся при запуске сервера |

# ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Изм. | Номера листов (страниц) | | | | Всего листов (страниц) в документе | № документа | Входящий № сопроводительного документа и дата | Подпись | Дата |
| измененных | замененных | новых | аннулированных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |