**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

## Текст программы

## ЛИСТ УТВЕРЖДЕНИЯ

## RU.17701729.507600-01 12 01-1-ЛУ

Исполнитель
студент группы БПИ175
_____ / И. О. Балбин /
«____»_____ 2017 г.

**Москва 2017**

УТВЕРЖДЕН
RU.17701729.507600-01 12 01-1-ЛУ

# Программа для адаптации музыкального произведения под определенный стиль на основе машинного обучения

**Текст программы**

**RU.17701729.507600-01 12 01-1**

**Листов 18**

| | |
|---|---|
| *Подп. и дата* | |
| *Инв. № дубл.* | |
| *Взам. инв. №* | |
| *Подп. и дата* | |
| *Инв. № подл* | |

**Москва 2017**

## СОДЕРЖАНИЕ

## 1. ТЕКСТ ПРОГРАММЫ

# run.py

```
from app import app
```

# config.py

```
import os

basedir = os.path.dirname(__file__)


class Config(object):
    UPLOAD_FOLDER = basedir + '\\uploads'
```

# __init__.py

```
from flask import Flask

from config import Config


app = Flask(__name__)

app.config.from_object(Config)


from app import routes
```

# routes.py

```
from flask import render_template, request, flash, redirect, url_for, send_from_directory

from app import app

from app.wavNN import calc_new_music

from app.midiNN.remix import lets_do_it

from multiprocessing import Process

import os
```

```
ALLOWED_EXTENSIONS_MID = set(['mid'])
ALLOWED_EXTENSIONS_WAV = set(['wav'])
def allowed_file(filename, mode):
   if mode == 'MID':
      return '.' in filename and \
        filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS_MID
   else:
      return '.' in filename and \
        filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS_WAV



@app.route('/uploads/<filename>')
def uploaded_file(filename):
   return send_from_directory(app.config['UPLOAD_FOLDER'],
                       filename)


@app.route('/index', methods=['GET', 'POST'])
@app.route('/', methods=['GET', 'POST'])
def index():
   return render_template("index.html")


@app.route('/process_data/wav/', methods=['Get','POST'])
def wavadapt():
   if request.method == 'POST':
```

```
        # check if the post request has the file part
        if ('style' not in request.files) or ('music' not in request.files):
            return redirect(request.url)
        style = request.files['style']
        music = request.files['music']
        if style.filename == '' or music.filename == '':
            return redirect(request.url)
        if style and allowed_file(style.filename, 'wav') and music and allowed_file(music.filename, 'wav'):
            music.save(os.path.join(app.config['UPLOAD_FOLDER'], "music.wav"))
            style.save(os.path.join(app.config['UPLOAD_FOLDER'], "style.wav"))
            if os.path.isfile('{}/out.wav'.format(app.config['UPLOAD_FOLDER'])):
                os.remove('{}/out.wav'.format(app.config['UPLOAD_FOLDER']))
            p = Process(target=calc_new_music)
            p.start()
            return redirect(url_for('wait'))

    return render_template("wav.html")

@app.route('/process_data/mid/', methods=['Get','POST'])
def midadapt():
    if request.method == 'POST':
        if 'music' not in request.files:
            return redirect(request.url)
        music = request.files['music']
        if music.filename == '':
```

```
            return redirect(request.url)
        if music and allowed_file(music.filename, 'MID'):
            mode = request.form['style']
            music.save(os.path.join(app.config['UPLOAD_FOLDER'], "music.mid"))
            if os.path.isfile('{}/out.mid'.format(app.config['UPLOAD_FOLDER'])):
                os.remove('{}/out.mid'.format(app.config['UPLOAD_FOLDER']))
            p = Process(target=lets_do_it, args=(mode,))
            p.start()
            return redirect(url_for('wait'))

    return render_template("mid.html")

@app.route('/wait', methods=['GET', 'POST'])
def wait():
    if request.method == 'POST':
        if 'out.mid' in os.listdir(app.config['UPLOAD_FOLDER']):
            return redirect(url_for('uploaded_file',
                                    filename="out.mid"))
        if 'out.wav' in os.listdir(app.config['UPLOAD_FOLDER']):
            return redirect(url_for('uploaded_file',
                                    filename="out.wav"))
        return render_template("waitpage.html")
    return render_template("waitpage.html")
```

# train.py

```
import random


import ngram

import glob


from keras import Sequential

from keras.layers import LSTM, Dense

from music21 import converter, instrument, note, chord, stream

from music21.ext import joblib

import music21

from sklearn.preprocessing import LabelBinarizer

from app import wild_card


#from heapq import nlargest

#import operator

import app.wild_card

from app.music_controller import get_msg, create_midi


def train(file):

    # Тут храним ноты

    notes = []

    # Счетчик для ограничения кол-ва мелодий

    nn = 0
```

```
print('Engage!')

# Тусуемся в папке

##for file in glob.glob("midi/mario/test/*.mid"):

   # Добавляем в хранилище еще ноты

notes.extend(get_msg(file))

   #nn+=1

   #if nn > 20:

      #break

# Сохраняем на всякий склеиную мелодию

#create_midi(notes)

print(notes)

# exit()

print("Создаем словарь всех возможных нот...")

encoder = LabelBinarizer()

encoder.fit(notes)

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

data = encoder.transform(notes)

print(len(data[0]))


print("Создаем ngram для поиска наиболее похожих нот...")

notes_set = set(notes)

ngram_notes = list(notes_set)

G = ngram.NGram(ngram_notes)

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

```
print("Создаем последовательности нот для обучения...")

look_back = 3

dataX, dataY = wild_card.create_dataset(data, look_back)


print("Перемешиваем мелодии...")

combined = list(zip(dataX, dataY))

random.shuffle(combined)

dataX[:], dataY[:] = zip(*combined)


print("Создаем тренировочные и тестовые данные...")

size_train = int(len(dataX)*0.8)

trainX = dataX[:size_train]

trainY = dataY[:size_train]


testX = dataX[size_train:]

testY = dataY[size_train:]


print('Создаем нейросеть...')

model = Sequential()

# model.add(LSTM(256, input_shape=(look_back, len(data[0])), return_sequences=True))

# model.add(Dense(256, activation='relu'))

# model.add(LSTM(256))

# model.add(Dense(256, activation='relu'))
```

```
# model.add(Dense(len(data[0])))
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'])

acc = []
print("Обучаем...")
for i in range(50):
    print("EPOCH " + str(i))
    out = model.fit(trainX, trainY, epochs=4, batch_size=32, verbose=1, shuffle=True)
    eval = model.evaluate(testX, testY, verbose=1)
    print(eval)
    acc.append(eval[1])
    #model.save("models/mario" + str(eval[1]) + ".h5")
    model.save("app/models/mario.h5")
#exit()
```

## wavNN.py

```
import librosa
import tensorflow as tf
import os
from IPython.display import Audio, display
import numpy as np
```

```
CONTENT_FILENAME = "uploads/music.wav"

STYLE_FILENAME = "uploads/style.wav"

OUTPUT_FILENAME = 'uploads/out.wav'


def calc_new_music():

    N_FFT = 2048

    def read_audio_spectum(filename):

        x, fs = librosa.load(filename)

        S = librosa.stft(x, N_FFT)

        p = np.angle(S)


        S = np.log1p(np.abs(S[:,:430]))

        return S, fs


    a_content, fs = read_audio_spectum(CONTENT_FILENAME)

    a_style, fs = read_audio_spectum(STYLE_FILENAME)


    N_SAMPLES = a_content.shape[1]

    N_CHANNELS = a_content.shape[0]

    a_style = a_style[:N_CHANNELS, :N_SAMPLES]


    N_FILTERS = 4096


    a_content_tf = np.ascontiguousarray(a_content.T[None,None,:,:])

    a_style_tf = np.ascontiguousarray(a_style.T[None,None,:,:])
```

```
# filter shape is "[filter_height, filter_width, in_channels, out_channels]"
std = np.sqrt(2) * np.sqrt(2.0 / ((N_CHANNELS + N_FILTERS) * 11))
kernel = np.random.randn(1, 11, N_CHANNELS, N_FILTERS)*std

g = tf.Graph()
with g.as_default(), g.device('/cpu:0'), tf.Session() as sess:
    # data shape is "[batch, in_height, in_width, in_channels]",
    x = tf.placeholder('float32', [1,1,N_SAMPLES,N_CHANNELS], name="x")

    kernel_tf = tf.constant(kernel, name="kernel", dtype='float32')
    conv = tf.nn.conv2d(
        x,
        kernel_tf,
        strides=[1, 1, 1, 1],
        padding="VALID",
        name="conv")

    net = tf.nn.relu(conv)

    content_features = net.eval(feed_dict={x: a_content_tf})
    style_features = net.eval(feed_dict={x: a_style_tf})

    features = np.reshape(style_features, (-1, N_FILTERS))
    style_gram = np.matmul(features.T, features) / N_SAMPLES
```

```
from sys import stderr


ALPHA= 1e-2

learning_rate= 1e-3

iterations = 100


result = None

with tf.Graph().as_default():


    # Build graph with variable input

    #x = tf.Variable(np.zeros([1,1,N_SAMPLES,N_CHANNELS], dtype=np.float32), name="x")

    x = tf.Variable(np.random.randn(1,1,N_SAMPLES,N_CHANNELS).astype(np.float32)*1e-3,
name="x")


    kernel_tf = tf.constant(kernel, name="kernel", dtype='float32')

    conv = tf.nn.conv2d(

      x,

      kernel_tf,

      strides=[1, 1, 1, 1],

      padding="VALID",

      name="conv")



    net = tf.nn.relu(conv)
```

```
content_loss = ALPHA * 2 * tf.nn.l2_loss(
       net - content_features)


style_loss = 0


_, height, width, number = map(lambda i: i.value, net.get_shape())


size = height * width * number
feats = tf.reshape(net, (-1, number))
gram = tf.matmul(tf.transpose(feats), feats)  / N_SAMPLES
style_loss = 2 * tf.nn.l2_loss(gram - style_gram)


# Overall loss
loss = content_loss + style_loss


opt = tf.contrib.opt.ScipyOptimizerInterface(
   loss, method='L-BFGS-B', options={'maxiter': 300})


# Optimization
with tf.Session() as sess:
   sess.run(tf.initialize_all_variables())


   print('Started optimization.')
   opt.minimize(sess)
```

```
        print ('Final loss:', loss.eval())
        result = x.eval()


a = np.zeros_like(a_content)
a[:N_CHANNELS,:] = np.exp(result[0,0].T) - 1


# This code is supposed to do phase reconstruction
p = 2 * np.pi * np.random.random_sample(a.shape) - np.pi
for i in range(500):
    S = a * np.exp(1j*p)
    x = librosa.istft(S)
    p = np.angle(librosa.stft(x, N_FFT))

librosa.output.write_wav(OUTPUT_FILENAME, x, fs)
```

# Приложение 1
## Список используемой литературы

1) ГОСТ 19.101-77 Виды программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
2) ГОСТ 19.102-77 Стадии разработки. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
3) ГОСТ 19.103-77 Обозначения программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
4) ГОСТ 19.105-78 Общие требования к программным документам. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
5) ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
6) ГОСТ 19.201-78 Текст программы. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
7) ГОСТ 19.604-78 Правила внесения изменений в программные документы, выполненные печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
8) ГОСТ 19.301-79 Программа и методика испытаний. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

# ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

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

| | | | | |
|---|---|---|---|---|
| Изм. | Лист | № докум. | Подп. | Дата |
| RU.17701729.507600-01 12 | | | | |
| Инв. № подл. | Подп. и дата | Взам. инв. № | Инв. № дубл. | Подп. и дата |