卒業研究報告書

題目

AI 生成楽曲を判別する AI の開発

指導教員

石水 隆　講師

報告者

19–1–037–0186

濱田 裕大

近畿大学理工学部情報学科

令和 6 年 2 月 4 日提出

概要

ChatGPT[1] などに代表される生成系 AI は，文章・画像・楽曲など様々なコンテンツを生成できる AI の総称である．ここ数年間で生成系 AI が生成する文章・画像・楽曲などの精度は大きく向上しており，テキストで指示するだけで、人間が制作した作品と判別しづらい文章や画像を作り出すことができる AI もある．

近年では生成系 AI を用いることにより，容易に文章，画像等を作成できるため，教育的な問題，倫理的な問題，法的な問題など様々な問題が発生している [2] [3]．学習に使用されるデータは原則，著作権者の許諾なしに利用可能なため，任意のクリエイターの作品を無断使用し，その特徴に似せて AI に出力させることもできてしまう．その結果，学習元への経済的還元が無かったり，模倣物と競合してしまったりすることに繋がる．楽曲の面では現在，いくつかの生成 AI モデルが，任意のアーティストの声や歌詞のテンポを学習し，本物と酷似したトラックを作成する機能を提供している． それらの問題に対処するために，生成系 AI が作成したものと，人間が作成したものとを区別できる手法の必要性が上がっており，「AI or Not」 [5] をはじめとした画像の判別するツールが開発されている．しかし，楽曲を判別するツールは現在確認できない．

そこで本研究では，AI 生成楽曲を判別などの生成系 AI をめざす．本研究では，AI 生成楽曲の特徴や人間が制作した楽曲との差などを調べ，その差を元に AI 生成楽曲であるかどうかを判別する AI を作成する．また，AI および AI 作成楽曲の利害やあり方についての考察を行う．

目次

1　序論

1.1　本研究の背景

ChatGPT[1] などに代表される生成系 AI は，文章・画像・楽曲など様々なコンテンツを生成できる AI の総称である．ここ数年間で生成系 AI が生成する文章・画像・楽曲などの精度は大きく向上しており，テキストで指示するだけで、人間が制作した作品と判別しづらい文章や画像を作り出すことができる AI もある．

近年では生成系 AI を用いることにより，容易に文章，画像等を作成できるため，教育的な問題，倫理的な問題，法的な問題など様々な問題が発生している [2] [3]．学習に使用されるデータは原則，著作権者の許諾なしに利用可能なため，任意のクリエイターの作品を無断使用し，その特徴に似せて AI に出力させることもできてしまう．その結果，学習元への経済的還元が無かったり，模倣物と競合してしまったりすることに繋がる．楽曲の面では現在，いくつかの生成 AI モデルが，任意のアーティストの声や歌詞のテンポを学習し，本物と酷似したトラックを作成する機能を提供している． それらの問題に対処するために，生成系 AI が作成したものと，人間が作成したものとを区別できる手法の必要性が上がっており，「AI or Not」 [5] をはじめとした画像の判別するツールが開発されている．しかし，楽曲を判別するツールは現在確認できない．そこで本研究では，AI 生成楽曲を判別などの生成系 AI をめざす．

1.2　本研究の目的

本研究では，AI 生成楽曲の特徴や人間が制作した楽曲との差などを調べ，その差を元に AI 生成楽曲であるかどうかを判別する AI を作成する．また，AI および AI 作成楽曲の利害やあり方についての考察を行う．

1.3　本報告書の構成

本報告書の構成は以下の通りである．まず第 2 章において本研究の内容について述べる．続いて第 3 章において本研究で作成した特徴検出プログラムについて説明する．第 4 章で結果と考察を述べ，最後に第 5 章でまとめと今後の課題を述べる．

2　研究内容

本研究では，楽曲生成 AI のひとつ AIVA [6] を用いて楽曲を生成し，それを音声認識 AI の学習データとすることで AI 生成楽曲が持つ特徴を検証する．機械学習等で使用するプログラムは，C++ 言語で作成する．また，AI が生成したものではない楽曲は，「フリー BGM（音楽素材）無料ダウンロード｜ DOVA-SYNDROME」 [7] や，「フリー BGM・音楽素材 MusMus」 [8] からダウンロードしたものを使用する．

2.1　楽曲を生成

AIVA 内で，Chord progression からランダムに楽器を選択し，1 分 30 秒〜2 分 30 秒程度の楽曲を 50 曲作成する．また，DOVA-SYNDROME と MusMus から人間が作曲した同じような長さの楽曲を 50 曲ダウンロードする．

2.2 楽曲データの読み取り

本研究では音声ファイルを読み取るためのプログラム (wavDecoder.cpp) を作成し，AIVA で作成したものと，DOVA-SYNDROME や MusMus でダウンロードしたものを，読み取り，データ化したものを buffer に格納する．

2.3 教師あり学習

楽曲の特徴を得て，教師あり学習でサポートベクトルマシンを用いて，AI と人間の 2 つのグループに分類するプログラム (svm.cpp) により機械学習を行う．サポートベクトルマシンとは，2 つのクラスに属する学習データ集合を，マージン (識別面と最も近い学習データとの距離) を最大化するように識別面を定め，線形分離する機械学習モデルのアルゴリズムである．[9]

2.4 楽曲を判別

学習に使用していない，新たな楽曲の音声データから特徴ベクトルを生成し，SVM モデルを使用して特徴ベクトルを分類することで，AI が生成したものかどうかを判別させる．

3 特徴検出プログラム

本章では，本研究で作成した楽曲の特徴抽出プログラムについて説明する．付録に本研究で作成した特徴抽出プログラムのソースを示す．ただし，現時点ではプログラムは未完成である．

以下に本研究で作成したプログラムについて説明する．

3.1 wavDecoder.cpp

PortAudio ライブラリにより音声ファイルを開きながら，while 文の中で buffer に保存し，情報をテキストファイル (output$_{d}ata.txt$) に行ごとに追記していく．1, 2の場合，「$a1.wav$」というファイルに対しての処理を行っているが, 9 行目を適切なファイル名に変更する．

3.2 svmr.cpp

ファイルから音声データを読み込み，新しい楽曲のデータを格納するためのベクトル「newMusicData」に格納する．データセットを準備した後，SVM モデルの設定し学習させる．

3.3 classfy.cpp

svm.cpp と同様に，ファイルから新たな音声データを読み込み，新しい楽曲のデータを格納するためのベクトル「newMusicData2」に格納する．mlpack::svm::SVM¡¿ svm; で学習済みモデルを読み込み，int 型の prediction に分類の値が入る．

4　結果と考察

現状，前章 2.2 での楽曲情報のデータ化及び，前章 2.2 でデータ化したものを 2.3 で利用する流れが繋がっておらず，2 つのグループに分類するプログラム (svm.cpp) を動作させられていない．その結果，楽曲を生成 AI のものであるかどうかを判別するまでに至らなかった．

考察としては，buffer に格納された音楽データを特徴ベクトルとして使用し，それに対応するクラスラベルを用意し各音楽データに対してクラス 1 またはクラス 2 のラベルを割り当てる部分に問題があったと考える．

つまり，svm.cpp のデータセットの部分特徴量とラベルを代入する部分を正確に記述することで，正常に動作するように改善されるだろう．

5　結論・今後の課題

それぞれボーカルありの楽曲で判別できるようにしたかったが，DOVA-SYNDROME や MusMus にアップロードされているボーカル有りの楽曲が少なく，学習データ不足になると考え，ボーカルの無い楽曲を用いての研究になった．判別の精度を向上させるには，学習データをさらに増やしていくことが望ましいが，wavDecoder.cpp の 8 行目のファイル名を手動で変更し，実行するという動作を何百回と行うことになってしまうため，その場合はファイル名を読み取る部分を for 文で自動化すべきである．また，プログラムをうまく実行できずエラーになってしまっていたことも多く，思うように研究を進めることができなかった．

謝辞

本研究を進めるにあたり，石水先生による温かい指導や支援をいただき，大変お世話になりました．至らない所が多い私に向き合ってくださり，本当にありがとうございました．

参考文献

1) ChatGPT, Open AI, https://chat.openai.com/
2) 生成ＡＩをめぐる議論，情報通信白書令和 5 年度版，総務省 (2023) , https://www.soumu.go.jp/
3) 武田俊之：大学は生成系 AI の影響をいかに認識しているか？，日本教育工学会 研究報告 Vol.2023, No.2, pp.88094 (2023), https://doi.org/10.15077/jsetstudy.2023.2_88
4) A.Sablayrolles, M.Douze, C.Schmid, H.Jégou： Radioactive data: tracing through training, arXiv Machine Learning, Cornell University (2020), https://arxiv.org/abs/2002.00937
5) AI or Not, https://www.aiornot.com/
6) AIVA, https://www.aiva.ai/
7) DOVA-SYNDROME, https://dova-s.jp/
8) MusMus, https://musmus.main.jp/
9) 荒井秀一：ビジュアルテキスト パターン認識，森北出版，2021

付録 A　ソースプログラム

本研究で作成したプログラムのソースファイルを以下に示す.

A.1　wavDecoder.cpp

```cpp
#include <iostream>
#include <vector>
#include <fstream>
#include <portaudio.h>
#include <sndfile.h>

#define SAMPLE_RATE 44100
#define FRAMES_PER_BUFFER 1024
#define FILE_NAME "a1.wav"

int main() {
    // 初期化
    Pa_Initialize();
    SNDFILE *sndfile;
    SF_INFO sfinfo;
    PaStream *stream;
    PaError err;

    // 音声ファイルを開く
    sndfile = sf_open(FILE_NAME, SFM_READ, &sfinfo);
    if (!sndfile) {
        std::cerr << "Error: Couldn't open the file " << FILE_NAME << std::endl;
        return 1;
    }

    // PortAudioの設定
    PaStreamParameters inputParameters;
    inputParameters.device = Pa_GetDefaultInputDevice();
    inputParameters.channelCount = sfinfo.channels;
    inputParameters.sampleFormat = paFloat32;
    inputParameters.suggestedLatency = Pa_GetDeviceInfo(inputParameters.device)->defaultLowInputLatency;
    inputParameters.hostApiSpecificStreamInfo = nullptr;

    // PortAudioのストリームを開く
    err = Pa_OpenStream(&stream, &inputParameters, nullptr, SAMPLE_RATE, FRAMES_PER_BUFFER, paClipOff, nullptr, nullptr);
    if (err != paNoError) {
        std::cerr << "Error: Couldn't open PortAudio stream" << std::endl;
        return 1;
    }

```

図 1　wavDecoder.cpp(1)

A.2　svm.cpp

A.3　classify.cpp

```cpp

    // ストリームを開始
    err = Pa_StartStream(stream);
    if (err != paNoError) {
        std::cerr << "Error: Couldn't start PortAudio stream" << std::endl;
        return 1;
    }

    // バッファの確保
    float buffer[FRAMES_PER_BUFFER * sfinfo.channels];

    // 音声データの読み取りと処理
    while (sf_readf_float(sndfile, buffer, FRAMES_PER_BUFFER) > 0) {
        // データをファイルに保存する
        std::ofstream outputFile("output_data.txt", std::ios::app); // 出力ファイルを追記モードで開く
        if (outputFile.is_open()) {
            for (int i = 0; i < FRAMES_PER_BUFFER * sfinfo.channels; ++i) {
                outputFile << buffer[i] << " ";
            }
            outputFile << "\n"; // データの区切りを示すために改行を追加
            outputFile.close(); // ファイルを閉じる
        } else {
            std::cerr << "Error: Couldn't open the output file" << std::endl;
            return 1;
        }

        // データの表示
        for (int i = 0; i < FRAMES_PER_BUFFER * sfinfo.channels; ++i) {
            std::cout << buffer[i] << " ";
        }
    }

    // ストリームと音声ファイルを閉じる
    Pa_CloseStream(stream);
    sf_close(sndfile);

    // 終了処理
    Pa_Terminate();

    return 0;
}
```

図 2　wavDecoder.cpp(2)

```cpp
#include <iostream>
#include <vector>
#include <fstream>
#include <mlpack/core.hpp>
#include <mlpack/methods/svm/svm.hpp>

int main() {
    // ファイルから音声データを読み込み、newMusicDataに格納
    std::vector<float> newMusicData;
    std::ifstream inputFile("output_data.txt");
    if (inputFile.is_open()) {
        std::string line;
        while (std::getline(inputFile, line)) {
            std::istringstream iss(line);
            float value;
            while (iss >> value) {
                newMusicData.push_back(value);
            }
        }
        inputFile.close();
    } else {
        std::cerr << "Error: Couldn't open the input file" << std::endl;
        return 1;
    }

    // データセットの準備
    std::vector<std::vector<float>> features = /* データセットの特徴量 */;
    std::vector<int> labels = /* データセットのラベル */;

    // SVMモデルの設定
    mlpack::svm::SVM<> svm(features, labels);

    // SVMモデルの学習
    svm.Train();

    return 0;
}
```

図 3　svm.cpp

```cpp
#include <iostream>
#include <vector>
#include <fstream>
#include <mlpack/core.hpp>
#include <mlpack/methods/svm/svm.hpp>

int main() {
// 新しい楽曲のデータを読み込む
    std::vector<float> newMusicData2;

    // ファイルから音声データを読み込み、newMusicData2に格納
    std::ifstream inputFile("new_music_data.txt");
    if (inputFile.is_open()) {
        float value;
        while (inputFile >> value) {
            newMusicData2.push_back(value);
        }
        inputFile.close();
    } else {
        std::cerr << "Error: Couldn't open the input file" << std::endl;
        return 1;
    }

    // 学習済みモデルを読み込む
    mlpack::svm::SVM<> svm;

    // 新しいデータを分類する
    int prediction = svm.Classify(newMusicData2);

    // 分類結果の表示
    std::cout << "Predicted class: " << prediction << std::endl;
    return 0;
}
```

図 4　class.cpp