平成 30 年度　卒業論文

AI を用いた音楽生成の研究

函館工業高等専門学校　生産システム工学科・情報コース
25 番　檜山　凜子

目次

第 1 章　序論

第 1 節　英文アブストラクト

It is a purpose to generate music using deep learning which is one of AI techniques. In this research, development was done in c++ environment using Visual Studio. Tiny-DNN was used as the AI library. Create training data in the CSV file, load it and do learning. Similarly, output can be done with a CSV file. In conclusion, although I was able to produce music, it is not accurate and I have to think about improving accuracy. Lack of training data and input perceptron is considered as a cause.

Key word：AI, deep learning, Tiny-DNN

第 2 節　研究目的

本研究では、音楽の知識が無い初心者による作曲を可能とさせることが目的である。そのため、手軽に作曲を可能とするツールの作成を試みる。

第 3 節　研究背景

昨今の情報社会において、AI の発展・活躍は目覚しいものがある。AI 技術への関心・理解を深めるため、ニューラルネットワークを用いた音楽生成をテーマとして研究を進めることとした。

第 4 節　開発環境

使用 PC：

OS：Windows 10 Education

CPU：Intel Core2 Duo E7400 @2.80GHz

実装 RAM：4.00GB

開発言語：C++

開発環境：Visual Studio 2017

第 2 章　音楽生成プログラムの開発

第 1 節　プログラムの概要

本プログラムでは、AI に教師データを与えて学習を行わせる。その後、与えた入力より学習に基づいた楽曲データの出力を試みた。このプログラムでは、ウェブ上で無料公開されている AI ライブラリ「Tiny-DNN」を使用し、開発環境は C++とした。教師データには任意の楽譜を数値に起こしたものを使用。楽譜内の音符の音階を MIDI ノート番号、符の長さを 1〜8 のノート番号としてサビにあたる部分を抜き出して表記した。

プログラムの流れは、はじめに CSV ファイル形式の教師データを読み込む。読み込んだデータを構築したニューラルネットワークに与えて学習させる。次いでプログラムが学習済みネットワークを利用してデータを出力。最後に出力したデータを CSV ファイルで保存する、という手順になっており、本プログラムには生成した音楽を再生する機能は無く、音楽の再生にあたっては音楽生成プログラムとは別に、音楽再生プログラムを作成した。

第 2 節　教師データと出力データ

本研究では教師データおよび出力データを CSV ファイル形式とした。教師データには、入力パーセプトロンとして BPM と作曲者識別番号を取り上げ、出力として任意の楽曲の楽譜を数字に起こしたものを起用した。楽譜の数字化は音階を MIDI ノート番号、音価を 1～8 のノート番号として表記することで実現した。一般的な楽曲に使われる音階は MIDI ノート番号上では 50～80 の数値になっており、音階・音価共に数字で表す点から AI の誤認率を下げるために音価には 1 桁台の小さな数字を起用した。教師データで使用した音階と MIDI ノート番号の対応例を表 1、音価に振ったノート番号を表 2 として次に示す。

表 1. 譜面の音階と MIDI ノート番号の対応例

音階	真中のド	レ	ミ	ファ	ソ
MIDI ノート番号	60	62	64	65	67

表 2. 音価に振ったノート番号

音価	1 分	2 分	付点 2 分	4 分	付点 4 分	8 分	付点 8 分	16 分
番号	1	2	3	4	5	6	7	8

本研究で AI に与えた教師データは、楽曲のすべてではなくサビ部分に相当するパートのみとした。理由としては、教師データの作成効率が悪く、一曲を全て作成するよりも一定のパートのみを抜き出しより多くの教師データを作成した方が良いと考えたためである。今回作成した教師データは 2 人の作曲者からそれぞれ 4 つずつの計 8 つであり、似通った楽曲を複数作成している作曲者を選ぶことで、AI にその特徴を掴ませることができるのではないかと考えた。

第 3 節　Tiny-DNN

本研究ではウェブ上で無料公開されている AI ライブラリ「Tiny-DNN」を使用した。Tiny-DNN はヘッダーファイルのみの構成であり、C++14 をサポートしているコンパイラが存在していればどのような環境でも対応できる。したがって、現開発環境に最も適していると判断し、Tiny-DNN を本研究の AI ライブラリとして導入した［1］。

第 4 節　ニューラルネットワーク

本研究では、AI の学習にあたってニューラルネットワークを利用した。ニューラルネットワークとは、人間の脳神経のニューロンを数理モデル化したものの組み合わせである。本研究では入力層を 2、中間層を 100、出力層を教師データに合わせて 180 とした。以下にプログラム上でのネットワーク構築をプログラム 1 として次に示す。

プログラム 1. ネットワークの構築

```
tiny_dnn::network<tiny_dnn::sequential> net;

net << fc(2, 100)                //fc(入力数、中間層数)
    << tiny_dnn::tanh_layer();
net << fc(100, 188)              //fc(中間層数、出力数)
    << tiny_dnn::sigmoid_layer();

auto loss = net.get_loss<cross_entropy_multiclass>(input,output);
printf("¥nloss -> %lf¥n", loss);
adagrad opt;
net.fit<mse>(opt, input, output, 4, 1000);
//(サンプルの数・教師データの数、エポック数・学習回数)

loss = net.get_loss<cross_entropy_multiclass>(input, output);
printf("loss -> %lf¥n", loss);

float bpm = 0.0;
float no = 0.0;
/*入力を与える。この値によって出力が変化する*/
printf("入力 1>>");
scanf("%f" ,&bpm);
printf("入力 2>>");
scanf("%f" ,&no);
printf("¥n 入力値 : %.0f , %.0f¥n",bpm,no);
tiny_dnn::vec_t input2 = { bpm, no };
vec_t a = net.predict(input2);
```

第 3 章　音楽再生プログラム

第 1 節　プログラムの概要

本プログラムでは、音楽生成プログラムで作成した CSV ファイルを読み込み、内包されている MIDI ノート番号と音符のノート番号をそれぞれ midiOut 関数と sleep 関数に当てはめて利用している。

音楽再生プログラムの流れは、始めに楽曲データである CSV ファイルを読み込む。次いでファイル内の MIDI ノート番号を midiOut メッセージに変換、音符のノート番号を適当なミリ秒に変換する。最後に変換した midiOut メッセージを関数に適用して鍵盤を押す、対応するミリ秒を sleep 関数に当てはめて状態を維持、再び midiOut 関数で鍵盤を離す。これらを繰り返すことによって、曲を奏でることを可能としている［2］。

第 2 節　midiOut 関数

本プログラムでは、生成した楽曲データを再生するにあたり midiOut関数を使用した。midiOut 関数は Windows API の一つであり、開発環境が Windows であったことから導入コストの低さを重視して起用した。midiOut 関数は、関数に固有のノート番号を直で打ち込むことで特定の鍵盤を押す・離すといった動作を実現することができ、またピアノやギターなどの音色も複数定義することが可能である。関数の性質上 wave ファイルや MIDI ファイルなどに変換することは不可能なため、プログラム上で楽曲データを音楽ファイルとして保存することはできない。出力されたデータを音楽ファイルとして保存する際には、音楽再生プログラムを実行中に PC 上の音を録音することで実現している。

第 3 節　MIDI ノート番号と音符ノート番号の変換

本プログラムでは、読み込んだ CSV ファイル内の MIDI ノート番号と音符ノート番号の変換をそれぞれ関数化した。

midiOut 関数で音を鳴らすためには、鍵盤を押す、止める、離す、以上の 3 ステップを必要とする。鍵盤を押すとき、プログラム上では midiOutShortMsg(h, 0x7f4390) 等と記述する。90 は鍵盤を押すこと、0x7f はベロシティ(鍵盤を押し込む速さ)を表し、43 は midiOut メッセージを表す。一方 CSV ファイル上では MIDI ノート番号と音符ノート番号のみが記載されているため、読み込んだデータを midiOut 関数の引数に適した値に変換する必要がある。この変換作業は switch 分を用いて、出力データに存在する値の範囲のみ実装した。

また、音符のノート番号をミリ秒変換するにあたり、CSV ファイルに記載している楽曲の BPM を参考とした。ミリ秒変換では、AI が予想外の値を返すことを想定し、予期しない数値の場合にはその秒数を 0 ミリ秒とした。

MIDI ノート番号の変換をプログラム 2、音符ノート番号のミリ秒変換をプログラム 3 として次に示す［3］。

プログラム 2. MIDI ノート番号変換関数

```
int midi(int* v) {
    switch(save) {
        case 84:
            *v = 0x7f5490;  // 鍵盤を押す C7 0x54(84) 127 チェンネル 0
            break;
        case 83:
            *v = 0x7f5390;  // 鍵盤を押す B6 0x53(83) 127 チェンネル 0
            break;
        case 82:
            *v = 0x7f5290;  // 鍵盤を押す A+,B-6 0x52(82) 127 チェンネル 0
            break;
        case 81:
            *v = 0x7f5190;  // 鍵盤を押す A6 0x51(81) 127 チェンネル 0
            break;
        case 80:
            *v = 0x7f5090;  // 鍵盤を押す G+,A-6 0x50(80) 127 チェンネル 0
            break;
        case 79:
            *v = 0x7f4f90;  // 鍵盤を押す G6 0x4f(79) 127 チェンネル 0
            break;

         ..........

        case 95:
            *v = 0x7f5f90;   // 鍵盤を押す F+,G-6 0x4e(78) 127 チェンネル
0
            break;
        default:
            *v = 0;
            break;
    }
    return *v;
}
```

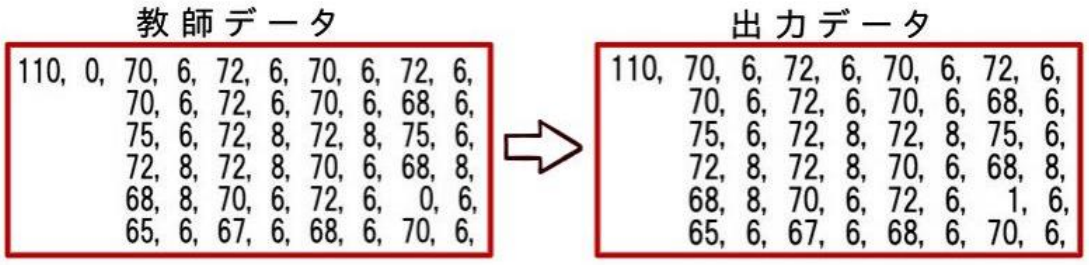
プログラム 3. 音符ノート番号のミリ秒変換関数

```
int msc(int* b) {
    switch (pop) {
        case 1:
            *b = (60000 / bpm) * 4;
            break;
        case 2:
            *b = (60000 / bpm) * 2;
            break;
        case 3:
            *b = (60000 / bpm) * 2.5;
            break;
        case 4:
            *b = (60000 / bpm) ;
            break;
        case 5:
            *b = (60000 / bpm) * 1.5;
            break;
        case 6:
            *b = (60000 / bpm) * 0.5;
            break;
        case 7:
            *b = (60000 / bpm) * 0.75;
            break;
        case 8:
            *b = (60000 / bpm) * 0.25;
            break;
       case 9:
            *b = (60000 / bpm) * 0.25;
            break;
        default:
            *b = 0;
            break;
    }
    return *b;
}
```

第 4 章　結果

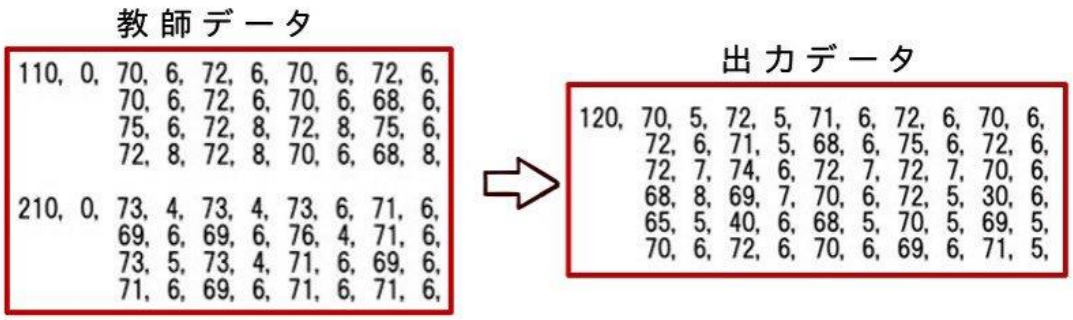
楽曲生成プログラムにおいて、始めに 1 つの教師データを AI に与え、出力としてほぼ同じデータを作成することに成功した。以下に図 1 として示す。

図 1. 教師データ 1 つの時の実行結果



次に、BPM の値が大きく離れた 2 つの教師データを与え、入力の BPM を教師データの片方に寄せて指定した。この際、寄せた方の教師データに近しい値を内包したデータが出力された。以下に図 2 として示す。

図 2. 教師データ 2 つの時の実行結果



さらに 4 つの教師データを与えた AI に対して、BPM として 60,120,180、作曲者識別番号として 0,0.5,1 をそれぞれ指定してデータを作成した。作曲者識別番号として与えた 0.5 は、出力時に 2 人の作曲者の特徴を織り交ぜたデータを作成させることが目的である。 出力結果の例として、BPM120・作曲者番号 0 とした時の出力結果を図 3、BPM120・作曲者番号 0.5 とした時の出力結果を図 4 として次に示す。

図 3. 教師データ 4 つ・BPM120・作曲者番号 0 の時実行結果

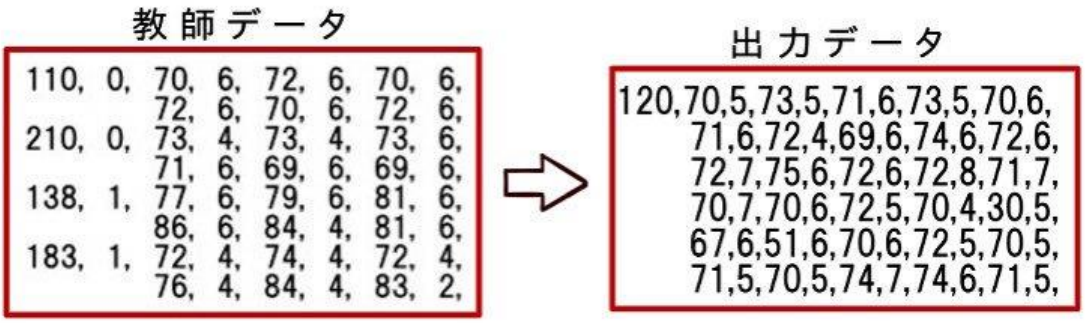
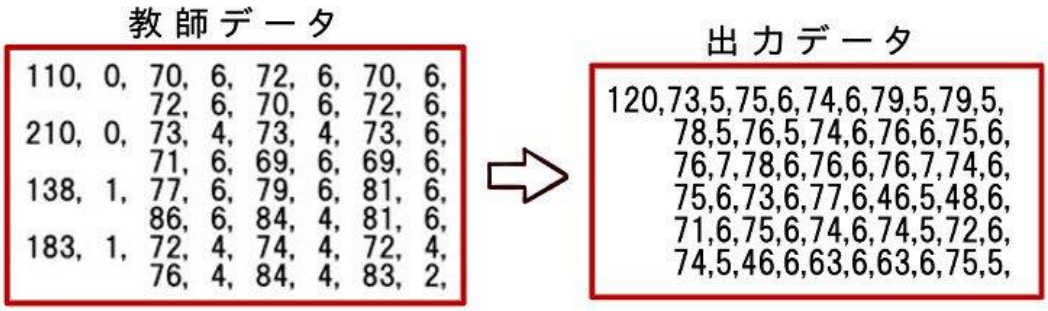


図 4. 教師データ 4 つ・BPM120・作曲者番号 0.5 の時実行結果



第 5 章　まとめ

結果より、AI が楽曲出力時に教師データをどのように利用しているかが凡そ理解することができた。入力として BPM を指定する際、指定された BPM に近しい値を持つ教師データを参照して、総合的な値を作成していると考えられる。結果の図 2 より、BPM を 120 に指定した出力データには、教師データ BPM110 側の要素が強く反映されており、210 側のデータはあまり参照していないことがわかる。同様に 4 つの教師データを与えた際も、入力パラメータの値がより近しいものを中心として総合的に値を出力していることが見て取れる。また、作曲者識別番号を 0.5 とした時、それぞれの作者の特徴を織り交ぜたデータを出力させることが目的であったが、総合的に値を見させた影響によって、規則性が見受けられない楽曲データが生成されてしまった。意図したデータを AI に作成させることができなかった原因としては、教師データや入力パーセプトロンの不足が考えられる。

以上より、現時点ではオリジナルの楽曲作成は非常に難しく、学習や学習させるデータがまだ不十分であると思われる。

第 6 章　課題

本研究における今後の課題として、より精度の高いデータ生成と和音の実装の 2 つがあげられる。

本研究では研究を始めるにあたって、AI 作曲技術についての事前知識を調べており、その段階で AI による作曲に膨大な教師データが必要であることが判明していた。現時点では教師データの作成が遅れデータ総数が 8 つと非常に少ない。また、入力として現段階では BPM・作曲者識別番号の 2 つを与えているが、これだけでは AI 側で認識できる事柄が圧倒的に少ないといえる。音楽知識のない初心者でも手軽に望ましい楽曲を生成する、という本研究の目的を果たすためには、利用者側に足りない分の知識を AI 側で賄う必要があると思われる。即ち、教師データを増やすことと入力パーセプトロンを増やすべきであると考えられる。

また、一般的にあらゆる楽曲において和音は多用されている。そのため、和音の実装ができなければ作曲データのクオリティ向上も難しいと考えられた。本研究では、研究過程において和音の実装を試みたが、学習が失敗したため時間コストを考慮して断念することとなった。この際、和音の教師データはこれまでの「MIDI ノート番号,音符データ」の順ではなく「MIDI ノート番号,MIDI ノート番号,音符データ」のように、2 音による和音であれば先に該当する 2 音の MIDI ノート番号を記述、後に音符データ。3 音であれば先に 3 音の MIDI ノート番号を記述、後に音符データ、という形式をとっていた。これを原因として、MIDI ノート番号との混同を避けて小さな値にしていた音符データのノート番号が意味をなさなくなり、出力データは和音ではなく出鱈目な数値の羅列となってしまった。今後の課題として、教師データの記述を改善し、より効率的に作成できる環境を整える必要があると考えられる。

参考文献

[1]GitHub -tiny-dnn
https://github.com/tiny-dnn/tiny-dnn

[2]MIDI による簡単な音の発生
http://yamatyuu.net/computer/program/vc2013/midi/index.html

[3]MIDI を鳴らす
http://www13.plala.or.jp/kymats/study/MULTIMEDIA/midiOutShortMsg.html