

ニューラルネットワークによる 音色の自動変換

教養学部後期課程学際科学科総合情報学コース

陶山大輝

学籍番号：08-192021

指導教官：金子知適准教授

目次

	Page
第 1 章 はじめに	5
第 2 章 背景：音	6
2.1 音の定義	6
2.1.1 音響信号	6
2.1.2 楽音	6
2.2 音の表現	7
2.2.1 一次元データと二次元データ	7
2.2.2 STFT	7
2.2.3 CQT	8
第 3 章 背景：ニューラルネットワーク	9
3.1 教師あり学習	9
3.1.1 教師あり学習の目的	9
3.1.2 教師あり学習モデルの学習と汎化	9
3.2 MLP	9
3.2.1 人工ニューロンの構造と定式化	9
3.2.2 MLP の構造	10
3.2.3 MLP の定式化	10
3.2.4 MLP の学習	11
3.3 CNN	12
3.3.1 CNN と視覚野の関係	12
3.3.2 疊み込み層	12
3.3.3 プーリング層	13
3.4 GAN	13
3.4.1 GAN の学習	13
3.4.2 GAN の定式化	13
3.5 Pix2pix	14
3.5.1 Pix2pix の定式化	14
3.5.2 Pix2pix の生成モデル	15
3.5.3 Pix2pix の識別モデル	15
第 4 章 提案手法	16
4.1 提案手法の目標	16
4.2 音の表現	16

4.3 提案モデル	16
4.3.1 生成モデル	17
4.3.2 識別モデル	17
第5章 実験	18
5.1 データセット	18
5.1.1 音階表記	18
5.2 実験方法	18
5.2.1 提案モデルの表現力の評価実験	18
5.2.2 提案モデルの汎化能力の評価実験	18
5.2.3 データ拡張	19
5.2.4 提案モデルの学習	19
5.2.5 評価方法	19
5.3 実験結果	19
5.3.1 概要：提案モデルの表現力の評価実験	19
5.3.2 概要：提案モデルの汎化能力の評価実験	20
5.3.3 課題	21
第6章 まとめ	23
6.1 実験結果のまとめ	23
6.2 展望	23
6.2.1 楽器の重ね合わせ	23
6.2.2 音の重ね合わせ	23
6.2.3 音の繋ぎ方	23
謝辞	24
参考文献	25
付録 A Adam	27
付録 B 学習時のパラメータ	28
付録 C データセットの分割	29
付録 D 実験結果	30
D.1 提案モデルの表現力の評価実験	30
D.2 提案モデルの汎化能力の評価実験	33

図目次

	Page
第 2 章	
図 2.1 音波	6
図 2.2 音波の拡大図	6
図 2.3 音響信号	7
図 2.4 STFT	7
図 2.5 メルスペクトログラム	8
図 2.6 CQT	8
図 2.7 Rainbowgram	8
図 3.1 MLP の人工ニューロン	10
第 3 章	
図 3.2 MLP のネットワーク	10
図 3.3 CNN の畠み込み層	12
図 3.4 CNN のプーリング層	13
図 3.5 GAN のネットワーク	13
図 3.6 Pix2pix のネットワーク	14
図 3.7 Pix2pix のスタイル変換の例	14
図 3.8 Pix2pix の生成モデルと識別モデル	15
第 4 章	
図 4.1 本研究の提案モデル	16
図 4.2 本研究の生成モデルと識別モデル	17
第 5 章	
図 5.1 88 鍵のピアノの鍵盤	18
図 5.2 F3 の音波	19
図 5.3 C4 の音波	19
図 5.4 D7♯ の音波	20
図 5.5 E7 の音波	20
図 5.6 D4♯ の音波	20
図 5.7 D7♯ の音波	20
図 5.8 D1 の音波	21
図 5.9 F6 の音波	21
図 5.10 A0 の音波	21
図 5.11 B7 の音波	21

図 5.12	C5 の音波	22
図 5.13	D2♯ の音波	22
図 5.14	F1♯ の音波	22
図 5.15	A7 の音波	22

第1章 はじめに

音楽は世界中で楽しまれ、その作成方法は技術発展により多様化している。近年注目されている音楽の作成方法としてリミックスと呼ばれるものがある。リミックスは既存の曲に音響操作を加えて新しい曲を作成することであり、1970年代のディスコの発達とともに世界的に普及した。当時はディスコでのDJによる即興のパフォーマンスにより行われていたが、近年のパソコンなどのデジタル機器の発達によって音楽の作成方法として一般的なものとなった。

しかし、録音や音の編集を行うソフトウェアアプリケーション (DAW) の操作がリミックスには必要であるため、音楽作成を円滑に行うためには一定の経験が必要となる。したがって、コンピュータプログラムによるリミックスの補助が音楽作成に役立つと考えられる。また、本研究では、リミックスの方法の一つである音色の変換に注目し、ニューラルネットワークによる変換手法を提案する。

さらに、音色の変換を行うためには、ある楽器の音を異なる楽器の音へ変換する技術が必要である。本研究では、画像のスタイル変換を行う Pix2pix [1] を音色の変換に応用した。Pix2pix は、ニューラルネットワークにより自然な画像を生成する手法である GAN [2] を応用した手法である。

そして、本研究では、ギターの単音をハープの単音へと提案モデルを用いて変換する実験を行った。その結果、ほとんどの音で音程を維持したまま音色の変換を行うことに成功した。また、データセットの一部の音のみで学習を行った場合でも、ほとんどの音で音程を維持したまま変換を行うことができた。

なお、本論文では、楽譜作成ソフトの MuseScore^{*1}を用いて音を作成し、音声編集ソフトの Audacity^{*2}を用いて音波の画像を作成した。

^{*1} <https://musescore.org/>

^{*2} <https://www.audacityteam.org/>

第2章 背景：音

本章では、音の定義を行い、音の表現方法を紹介する。

2.1 音の定義

音とは、弾性体中を伝播する波により起こされる音波が聴覚により感じられるもののことである。

2.1.1 音響信号

音は時間方向の変化量であるため、音響信号と呼ばれる。音響信号は連続的な信号であるが、コンピュータで扱うために離散的な信号へと変換する必要がある。また、変換の際には標本化（サンプリング）と量子化が必要である。まず、サンプリングは一定の時間を空けて離散的に測定を行うことであり、1秒あたりのサンプリング回数をサンプリング周波数と呼ぶ。そして、量子化は信号の大きさを離散的に表現することであり、信号の大きさを表現するビット数を量子化ビット数と呼ぶ。

2.1.2 楽音

楽音は周期性のある音波を持つ音のことであり、音楽で用いられる。本論文では楽音としての音を生成することを目標とする。また、楽音は長さ、大きさ、高さ、音色の四要素を持つ。

2.1.2.1 音の長さと大きさ

音の長さは音波の時間長により決まり、音の大きさは音波の振幅により決まる（図2.1）。音波の時間長が長いほど音の長さは長くなり、音波の振幅が大きいほど音の大きさは大きい。

2.1.2.2 音の高さと音色

音の高さと音色は直感的にはそれぞれ音波の周期構造の長さと形により決まる（図2.2）。また、音の高さは音波の周波数により決まり、周波数の高い音ほど音の高さは高くなる。そして、音の長さと大きさと高さが同じ時の音の違いを音色と呼び、それぞれの楽器は異なる音色を持つ。

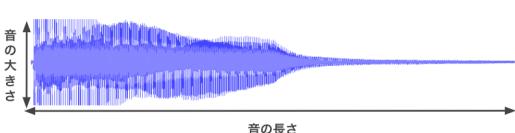


図 2.1: 音波

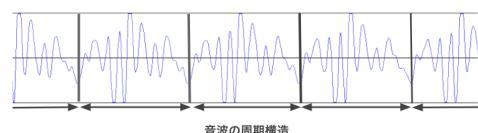


図 2.2: 音波の拡大図

2.2 音の表現

音をニューラルネットワークで扱うためには、楽音としての特徴を学習するための適切な表現を得る必要がある。また、本節は [3] の 3.2 節及び [4] の 2.1 節を参考に作成し、本節の図は [3] の Figure 4 と [4] の Figure 2 を利用している。

2.2.1 一次元データと二次元データ

音響信号をニューラルネットワークでは一次元データとして扱うが（図 2.3）、素朴な音の表現である一次元データから楽音としての特徴を学習するのは難しい。したがって、[5] のように階層型のネットワークを用いるなどの工夫が必要となる。また、一般的には音響信号を周波数成分ごとに分解して二次元データに変換することが多い。二つの次元としては時間と周波数を選ぶことがほとんどで、この場合の二次元データを時間-周波数表現と呼ぶ。

2.2.2 STFT

STFT (Short Time Fourier Transform) は、同じバンド幅の中間周波数を利用したバンドパスフィルターを用いて音響信号を周波数成分ごとに分解する手法である。また、STFT を用いて生成できる時間-周波数表現を可視化した画像をスペクトログラムと呼び、振幅の大きさを明度などにより表す（図 2.4）。ただし、スペクトログラムには位相の情報は保持されないため、音響信号からスペクトログラムへの変換は不可逆である。そして、離散信号に対しての STFT は式 2.1 により定式化される。ここで、 m は時刻、 ω_k は周波数、 w は 0を中心とした窓関数、 $x(n)$ は時間 n における信号 x の値、である。

$$\text{STFT}(m, \omega_k) = \sum_{n=-\infty}^{\infty} x(n)w(n-m)e^{-i\omega_k n} \quad (2.1)$$

また、STFT は人間の聴覚系の周波数分解能に合わせて作られたものではないため、楽音の解析の手法として用いる際はスペクトログラムに何らかの工夫を加えることが多い。周波数方向の圧縮方向を施した一般的なスペクトログラムとしてはメルスペクトログラムがあげられる（図 2.5）。メルスペクトログラムでは、人間の感じる音の高さの差が等幅になるように調整した尺度であるメル尺度 [6] を用いて周波数方向の圧縮を行う。

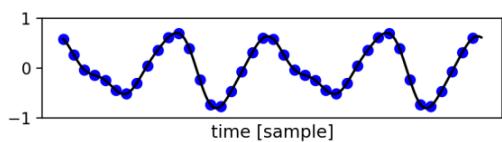


図 2.3: 音響信号

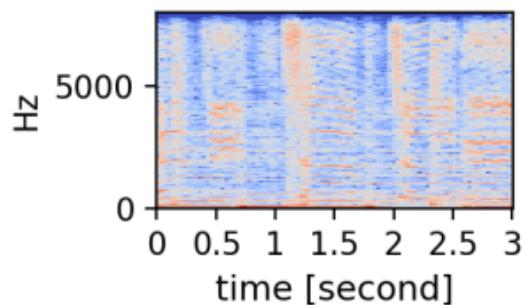


図 2.4: STFT

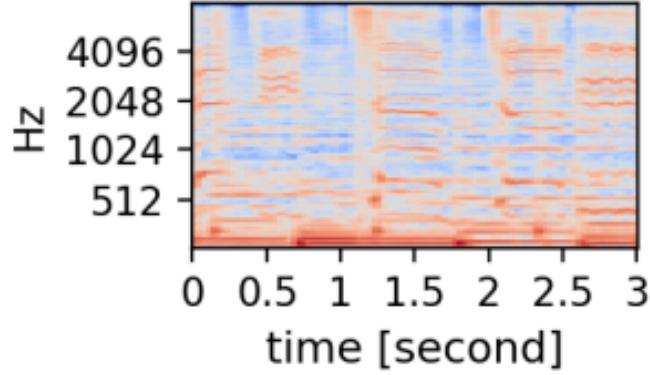


図 2.5: メルスペクトログラム

そして、メル尺度は、[7] では周波数を f として式 2.2 のように定式化される。メル尺度は人間の聴覚系に合わせた尺度であるため、メルスペクトログラムはスペクトログラムよりも楽音の解析に適している。

$$mel(f) = 1127.01048 \log\left(\frac{f}{700} + 1\right) \quad (2.2)$$

2.2.3 CQT

CQT (Constant Q Transform) [8] は STFT と同様に周波数成分ごとに分解する手法であるが、二つの工夫を行う。まず、定バンド幅のバンドフィルターを持つ STFT には低周波数成分のサンプル数が少なくなり分解能が低下するという問題がある。したがって、CQT では低周波数成分ほどバンド幅を広げることで分解能を一定にする工夫を行う。そして、人間の聴覚系の分解能は線形よりも対数に近いため、CQT では対数振幅を用いる工夫を行う。

また、CQTにおいては対数振幅を用いたスペクトログラムを生成することができる(図 2.6)。そして、位相の時間微分をカラーコードにより表現した Rainbowgram も音波の生成時の可視化による評価として [9] や [10] で用いられる(図 2.7)。

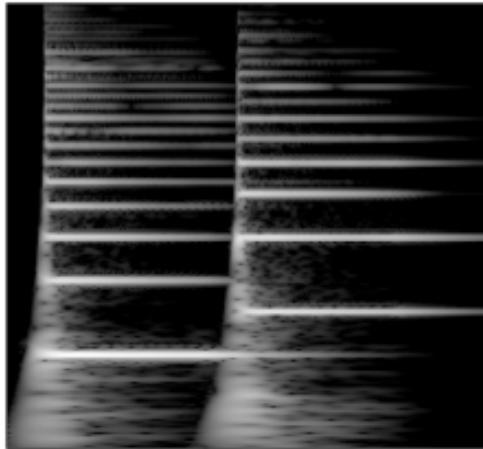


図 2.6: CQT

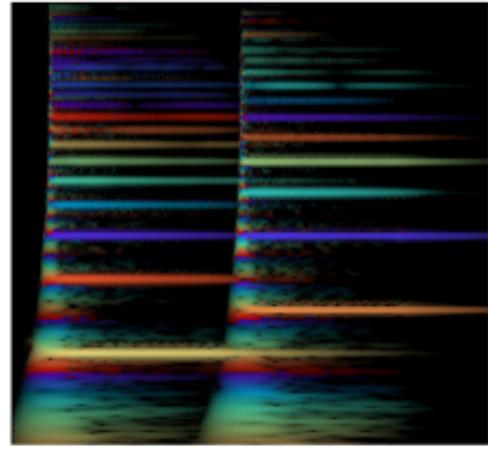


図 2.7: Rainbowgram

第3章 背景：ニューラルネットワーク

本章では、教師あり学習の説明を行った後、本論文で必要となるニューラルネットワークの説明を行う。また、本章の図では [1] の Figure 1 を利用している。

3.1 教師あり学習

教師あり学習とは、学習データとして説明変数と対応するべき目的変数のペアが与えられる機械学習の手法である。また、機械学習とは、学習データに含まれる特徴をコンピュータプログラム（モデル）が自動で学習し、学習したモデルを用いて何らかの問題を解く手法のことである。

3.1.1 教師あり学習の目的

X, Y をそれぞれ説明変数と目的変数の集合とすると、 $f : X \rightarrow Y$ のうち任意の $x \in X$ について正しい値 $y \in Y$ を出力する関数 f' を表現するモデルを作成することが教師あり学習の目的である。

また、 $f(x)$ の $f'(x)$ への近似の程度を評価する関数を損失関数 L と呼ぶ。損失関数 L は $L : Y \times Y \rightarrow \mathbb{R}^+$ として定義され、値が小さいほど近似の程度が良い関数である。 \mathbb{R}^+ は非負の実数を表す。

ここで、モデルのパラメータを θ とすると、教師あり学習は損失関数の期待値を最小化する θ' を求める最適化問題として考えられ、 θ' により決まる関数 f は f' となる。

3.1.2 教師あり学習モデルの学習と汎化

任意の x と対応する y の組を用意することは現実的には難しい。したがって、学習データのみでの最適化問題を解いて最適解として $\hat{\theta}$ を求めることが教師あり学習モデルの学習の目標である。

しかし、 $\hat{\theta}$ により定まる \hat{f} は f' に一致するとは限らない。従って、 \hat{f} の f' への近似の程度を評価する必要がある。そこで、学習データとは別に評価データを用意し、評価データについての損失関数の期待値を求めるなどにより未知のデータへのモデルの性能（汎化性能）を評価する。

3.2 MLP

MLP (Multilayer perceptron) は教師あり学習に用いられるニューラルネットワークである。また、ニューラルネットワークは神経細胞間のシナプス結合を通る電気信号により実現される脳の機能に類似した数理モデルの総称である。

3.2.1 人工ニューロンの構造と定式化

MLP では、神経細胞を図 3.1 のような人工ニューロンとして表現する。また、この人工ニューロンでは W_i を重みとした x_i の重み付き和にオフセットとして b を加えた値に活性化関数 θ を作用させた値を y として出力するため、 $y = \theta(\sum_{i=1}^m W_i \times x_i + b)$ として定式化される。

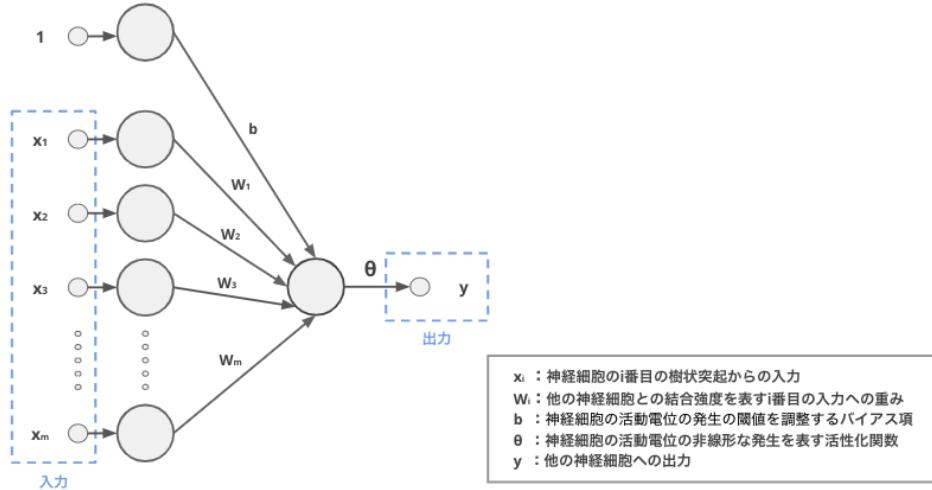


図 3.1: 人工ニューロン

3.2.2 MLP の構造

MLP は、図 3.1 のような人工ニューロンが複数並んだ層を一層とし(図 3.2a)、この層により順伝播型の階層構造のニューラルネットワークを形成する(図 3.2b)。また、層構造として入力層と中間層と出力層を持ち、入力層と出力層はそれぞれ一層のみである。ここで、層を構成する人工ニューロンの出力は次の層の全ての人工ニューロンに渡されるため、MLP を構成する層を全結合層と呼ぶ。

3.2.3 MLP の定式化

MLP の一層は式 3.1 として定式化され、MLP の全層は式 3.2 として定式化される。

$$y = \theta(Wx + b) \quad (3.1)$$

$$y = \theta_n(\theta_{n-1}(W_{n-1} \cdots (\theta_1(W_1x + b_1)) \cdots + b_{n-1}) + b_n) \quad (3.2)$$

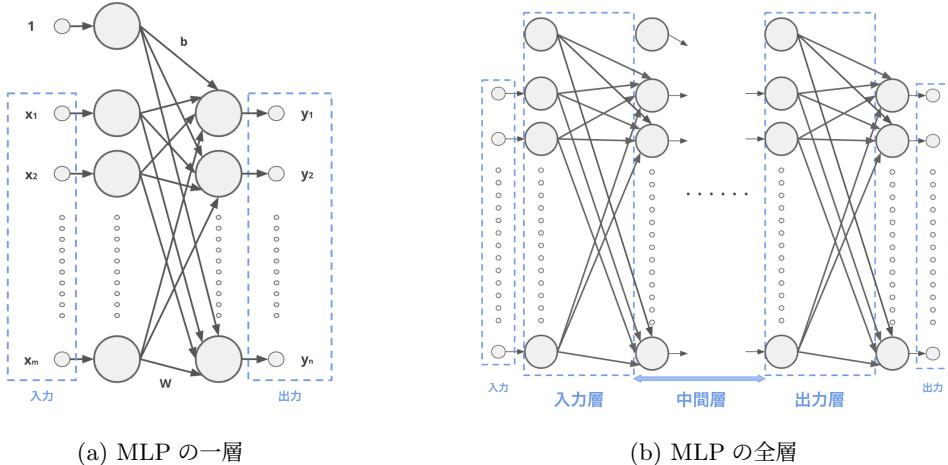


図 3.2: MLP のネットワーク

ここで、 $\mathbf{x}, \mathbf{y}, \mathbf{b}$ は第 i 成分が層の i 番目の人工ニューロンに対応するベクトルであり、それぞれ入力と出力とバイアス項に対応する。そして、 W は (i, j) 成分が層の j 番目の人工ニューロンの i 番目の出力の重みに対応する重み行列であり、活性化関数 θ はそれぞれの出力に対して同様に作用する。また、 n は層数であり、下付きの数字は何番目の層であるかを表す。

3.2.4 MLP の学習

MLP では、学習データを実ベクトルに変換した後に学習を行う。また、実行可能解はパラメータ W_i, \mathbf{b}_i であり、最適解を求めるために順次更新をする。この時、損失関数 L の勾配を用いた最適化アルゴリズム（勾配降下法）を用いて更新するのが一般的である。

3.2.4.1 勾配降下法

勾配降下法の中で最も単純なアルゴリズムである最急降下法を紹介する。最急降下法では、最も急な降下方向である勾配の反対方向へパラメータの更新を行う。具体的には、パラメータ W_i, \mathbf{b}_i はそれぞれ式 3.3 と式 3.4 にしたがって更新される。

$$W_i \leftarrow W_i - \eta \frac{\partial L}{\partial W_i} \quad (3.3)$$

$$\mathbf{b}_i \leftarrow \mathbf{b}_i - \eta \frac{\partial L}{\partial \mathbf{b}_i} \quad (3.4)$$

ここで、 η は学習率と呼ばれる正の実数であり、更新の程度を指定する。また、勾配降下法としては SGD (Stochastic Gradient Descent) や Adam (Adaptive Moment Estimation) などが一般には用いられる [11]。本研究で用いるアルゴリズムの Adam の挙動は付録 A に示す。

3.2.4.2 誤差逆伝播法

勾配降下法ではそれぞれのパラメータについて損失関数の勾配を求める必要があり、高速に勾配を求める方法として誤差逆伝播法が一般に用いられる。また、誤差逆伝播法は大きく二つの段階に分けることができる。一つ目の段階では、与えられた学習データを元に入力層から出力層の方向に（順伝播）それぞれの人工ニューロンの出力を求める。二つ目の段階では、出力層における損失関数の勾配を求めた後、勾配を誤差として出力層から入力層の方向に（逆伝播）伝えながら連鎖律を使用した勾配の計算をそれぞれのパラメータについて行う。また、誤差逆伝播法と連鎖律についてはそれぞれ [12] の 5.3 節と [13] の 4.4 節に詳しい。

3.2.4.3 バッチ処理

3.2.4.1 節における最急降下法を MLP で用いる場合、一回の更新ごとに全ての学習データの期待値として誤差関数の値を求める。このように全てのデータをひとまとめに扱う方法をバッチ処理と呼び、並列計算による高速化も行うことができる。しかし、一回の更新ごとに全データの勾配計算をするために計算効率が悪いかつメモリに収まらない場合は計算を行うことができない。そこで、SGD では一回の更新ごとに一つのデータをランダムに選ぶことでバッチ処理からの改善を狙った。しかし、SGD においてはその更新における最適解への収束が難しいだけでなく、並列計算による高速化も行うことができない。そこで、ミニバッチ処理と呼ばれる方法を採用するのが一般的である。ミニバッチ処理では、一回の更新ごとに指定したバッチサイズのサブセットのみを用いる。これにより、バッチ処理と SGD の両方の良い点を用いることができると期待できる。

3.3 CNN

CNN (Convolution Neural Network) は、MLP に全結合層だけでなく畳み込み層とプーリング層を加えた順伝播型のニューラルネットワークである。2012 年の ILSVRC で 2 位以下に 10% 以上の画像の認識精度の差をつけて優勝した AlexNet [14] はその著名な例である。また、本章では二次元データの画像において CNN の説明を行うが、詳しくは [15] を参考にされたい。

3.3.1 CNN と視覚野の関係

CNN は脳の視覚野の機能を模倣するニューラルネットワークであり、Neocognitron [16] を起源に持つ。また、脳の視覚野には主に二つの機能がある。一つ目は刺激の位置により異なった位置のニューロンの活動電位を発生させることであり、二つ目は視覚情報の特徴を段階的に受容することである。

上記の二つの機能を畳み込み層とプーリング層を用いて CNN は表現することができる。まず、畳み込み層では、人工ニューロン間の結合を局所領域に限定することで局所領域の特徴を抽出する。また、局所領域間の位置関係は変わらないため、前者の機能を模倣することができる。そして、プーリング層では、局所領域の人工ニューロンの出力をまとめることで局所領域の特徴を集約する。また、層を進むにつれて高段階の特徴が残るため、後者の機能を模倣することができる。

3.3.2 畳み込み層

畳み込み層においては、入力の行列に対してカーネルと呼ばれるフィルターを用いた畳み込み演算により行列を出力する(図 3.3)。カーネルは全結合層の重みに相当する値を持つ行列であり、ストライドと呼ばれる一定の間隔を空けながら上下方向にスライドする。また、畳み込み演算ではカーネルにより覆われた入力の部分に対して重み付き和の計算が行われる。そして、入力と出力の大きさを変えないために入力の周りに適当な幅で値を埋めるのが一般的であり、この操作をパディングと呼ぶ。

ここで、図 3.3 では 1 つの入力の行列に対し 1 つの出力の行列を返すが、これらの入出力で扱う行列を特微量マップと呼ぶ。そして、特微量マップの枚数を表現する際には特微量マップをチャンネルと呼ぶことが一般的であり、この畳み込み層は入出力が 1 チャンネルずつであると言える。また、入出力のいずれも複数チャンネルであることが多い。

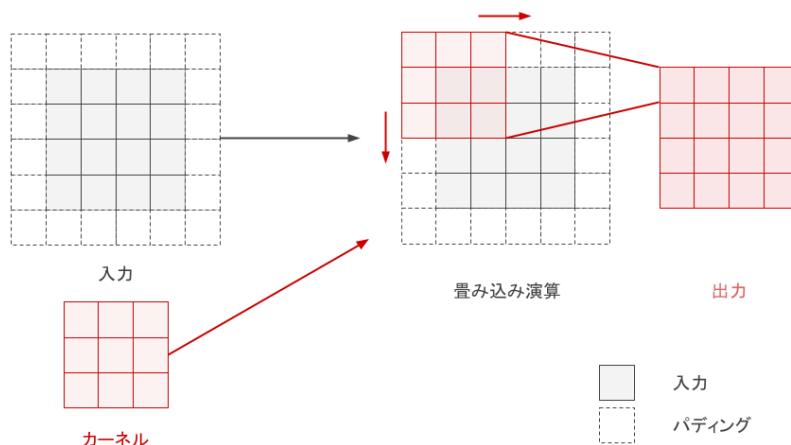


図 3.3: 畳み込み層

3.3.3 プーリング層

プーリング層においては、入力を局所領域ごとに分解した後に領域ごとへの何らかの操作により情報量の圧縮を行う（ダウンサンプリング）。また、この時の操作で最大値をとる場合は Max Pooling、平均値をとる場合は Average Pooling と呼び、主にこの二つが用いられる。

3.4 GAN

GAN (Generative Adversarial Networks) [2] はニューラルネットワークの応用例であり、学習データの特徴を持つ擬似的なデータを生成することを目指す手法である。この手法を用いると、実在しない人の写真やホテルの写真を生成することができる [17]。また、GAN は Discriminator (識別モデル) と Generator (生成モデル) の二つのニューラルネットワークで構成される（図 3.5）。

3.4.1 GAN の学習

生成モデルと識別モデルの二つのネットワークはランダムに初期化された後に競合的に学習を進める。まず、識別モデルはデータが Fake data (生成モデルの出力) と Real data (学習データ) のどちらであるかを識別できるように学習を進める。そして、生成モデルは識別モデルが学習データであると誤って識別するように、Noise (ノイズ) を元に学習データに近いデータを出力する。この二つの学習を交互に繰り返すことで、漸進的に生成モデルが学習データにより近いデータを生成できるようになると期待される。また、ノイズは適当な次元のベクトルであり、生成モデルの出力の揺らぎを表現する潜在変数の役割を果たす。

3.4.2 GAN の定式化

GAN では、生成モデルと識別モデルの目的関数はそれぞれ式 3.5、式 3.6 として定式化される。

$$\arg \min_{\theta_G} \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z}; \theta_G); \theta_D))] \quad (3.5)$$

$$\arg \max_{\theta_D} \mathbb{E}_{\mathbf{x}}[\log D(\mathbf{x}; \theta_D)] + \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z}; \theta_G); \theta_D))] \quad (3.6)$$

ここで、 \mathbf{x} は学習データ、 $G(\mathbf{z}; \theta_G)$ はノイズ \mathbf{z} を入力とする生成モデル、 $D(\cdot; \theta_D)$ は識別モデル、 θ_G と θ_D はそれぞれ生成モデル G と識別モデル D のパラメータ、である。

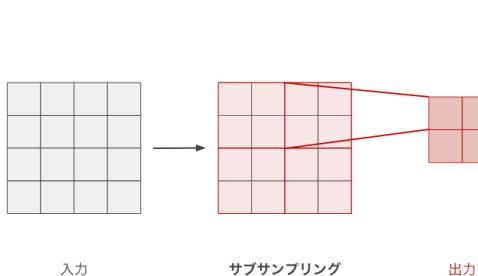


図 3.4: プーリング層

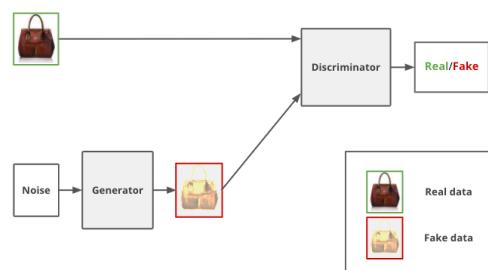


図 3.5: GAN のネットワーク

3.5 Pix2pix

Pix2pix [1] は、ネットワークの入力に変換元の画像を Condition (条件) として与えることで画像の変換を行う GAN である (図 3.6)。特定の条件をネットワークの入力に与える GAN としては Conditional GAN (CGAN) [18] が初めて考案されたが、Pix2pix は与えられた条件画像の構造を維持したまま変換するという点で CGAN とは異なる。具体的には、線画から写真への変換や白黒画像からカラー画像への変換を Pix2pix により行うことができる (図 3.7)。

3.5.1 Pix2pix の定式化

Pix2pix では、生成モデルと識別モデルの目的関数はそれぞれ式 3.7、式 3.8 として定式化される。

$$\arg \min_{\theta_G} \mathbb{E}_{y,z}[\log(1 - D(y, G(y, z; \theta_G); \theta_D))] + \mathbb{E}_{x,y,z}[\|x - G(y, z; \theta_G)\|_1] \quad (3.7)$$

$$\arg \max_{\theta_D} \mathbb{E}_{x,y}[\log D(x, y; \theta_D)] + \mathbb{E}_{y,z}[\log(1 - D(y, G(y, z; \theta_G); \theta_D))] \quad (3.8)$$

ここで、 x は変換先の学習データ、 y は変換元の学習データ、 z は生成モデルへの入力のノイズ、 $G(y, z; \theta_G)$ は y を条件としノイズ z を入力とする生成モデル、 $D(y, \cdot; \theta_D)$ は y を条件とする識別モデル、 θ_G は生成モデル G のパラメータ、 θ_D は識別モデル D のパラメータ、である。

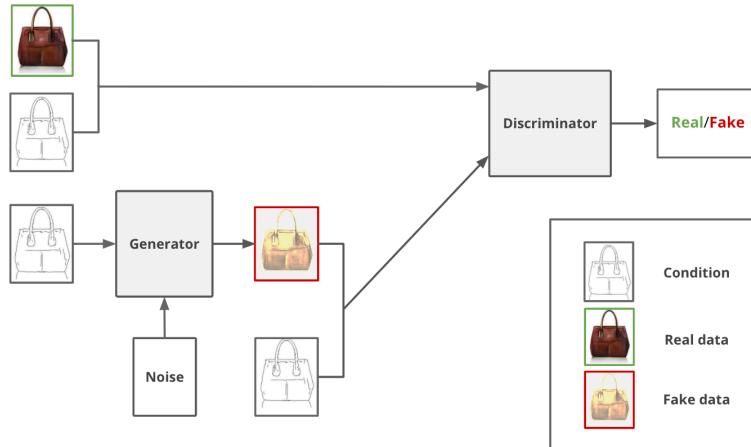


図 3.6: Pix2pix のネットワーク



Edges



Photo



BW



Color

(a) Edges to Photo

(b) BW to Color

図 3.7: Pix2pix のスタイル変換の例

3.5.2 Pix2pix の生成モデル

Pix2pix の生成モデルにはスキップコネクションを持った Encoder-Decoder を用いる (図 3.8a)。また、GAN のノイズとしてはノイズベクトルではなく Dropout を用いる。

3.5.2.1 Encoder-Decoder

Encoder-Decoder は、画像から特徴量を抽出する処理 (Encode) と特徴量を元に画像を再構成する処理 (Decode) の二段階で構成されたモデルである。また、Encode の部分には畳み込み層を用い、Decode の部分には逆畳み込み層を用いる。逆畳み込み層では畳み込み層の逆演算に相当する演算を行う、逆畳み込み層については [19] の 3.2.2 節に詳しい。

3.5.2.2 スキップコネクション

スキップコネクションは図 3.8a に示す赤線であり、変換元の画像と変換後の画像で共通する基本構造であるピクセルの対応関係を維持すると期待される。また、U-net [20] で用いられたものと同様のスキップコネクションであり、エンコード時の特徴量マップをデコード時にも利用する。

3.5.2.3 Dropout

Dropout [21] は、学習時にベルヌーイ分布に従ってランダムにある層の重みの一部を 0 として無視することで正則化を行う手法である。Pix2pix ではノイズベクトルを用いた際にノイズを無視して生成モデルが学習をするため、Dropout を用いる。

3.5.3 Pix2pix の識別モデル

Pix2pix の識別モデルには PatchGAN を用いる。PatchGAN は画像全体の真偽ではなく局所領域ごとの真偽の平均を識別に用いるため、局所的な識別精度が高まることが期待される (図 3.8b)。

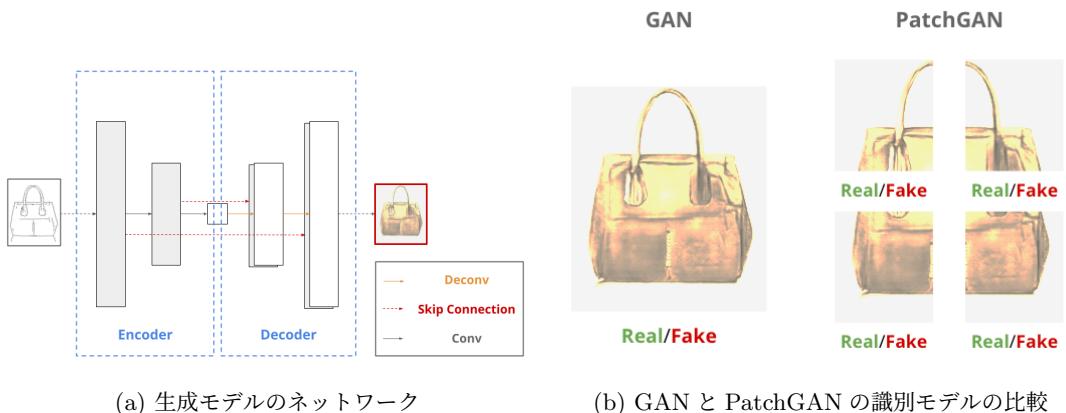


図 3.8

第 4 章 提案手法

本章では、提案手法の説明を行う。

4.1 提案手法の目標

提案手法の目標は、提案モデルとなるニューラルネットワークを用い、音の高さと大きさを維持したまままでギターの単音からハープの単音への音色の変換を行うことである。また、単音とはある楽器を用いてある高さの一音を鳴らした時に出力される音として定義する。ここで、ほとんどの単音は複数の周波数成分の合成波であり、音の高さは最も低い周波数成分の音（基音）の高さである。また、音色の違いは基音より高い音（上音）の音波の組み合わせの違いに起因する。

4.2 音の表現

44100 Hz のサンプリング周波数でサンプリングを行った 1 秒の音響信号を音の表現として使用した。また、量子化ビット数は 16 ビット、チャンネル数は 1 である。したがって、本研究で用いる音は 44100 の長さを持つ 16 ビット整数の一次元配列として表現される。

4.3 提案モデル

本研究では、Pix2pix [1] を元に生成モデルと識別のいずれにも条件として変換元の音を入力することで音色の変換を行う GAN を提案モデルとして作成した（図 4.1）。

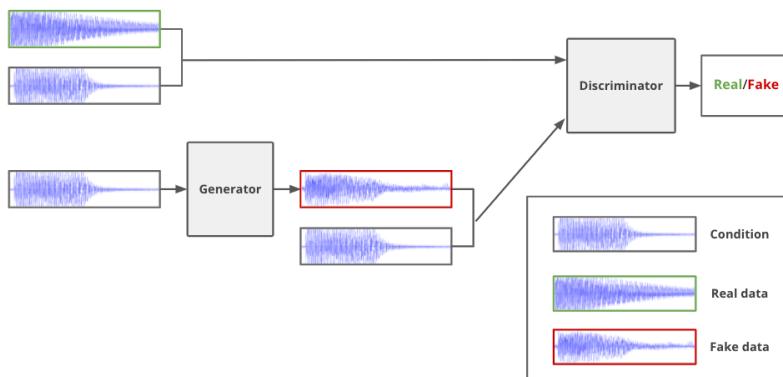


図 4.1: 提案モデル

図 4.2 は [20] の Figure 1 を参考に作成した。灰色の箱は特微量マップであり、箱の上側にチャンネル数を示し、箱の左側に特微量マップとなる一次元配列の長さを示す。また、ネットワーク構造の下の囲みの中にそれぞれの矢印が対応するニューラルネットワークの操作を示す。畠み込み層と逆畠み込み層については (カーネルサイズ、パディング数、ストライド) としてそれぞれの値を示す。活性化関数である LeakyReLU については負の実数の定義域での一次関数の傾きの値を示す。

4.3.1 生成モデル

生成モデルには、Pix2pix [1] を元に 1 つのスキップコネクションを持つ Encoder-Decoder 型のネットワークを用いた。また、条件となる変換元の音波を生成モデルの入力とする (図 4.2a)。また、このモデルでは決定論的に音を生成するために生成モデルの入力にノイズを使用していない。

4.3.2 識別モデル

識別モデルには、Pix2pix [1] を元に一般的な CNN を用いた。また、最終層の特微量マップの平均値を出力とする (図 4.2b)。

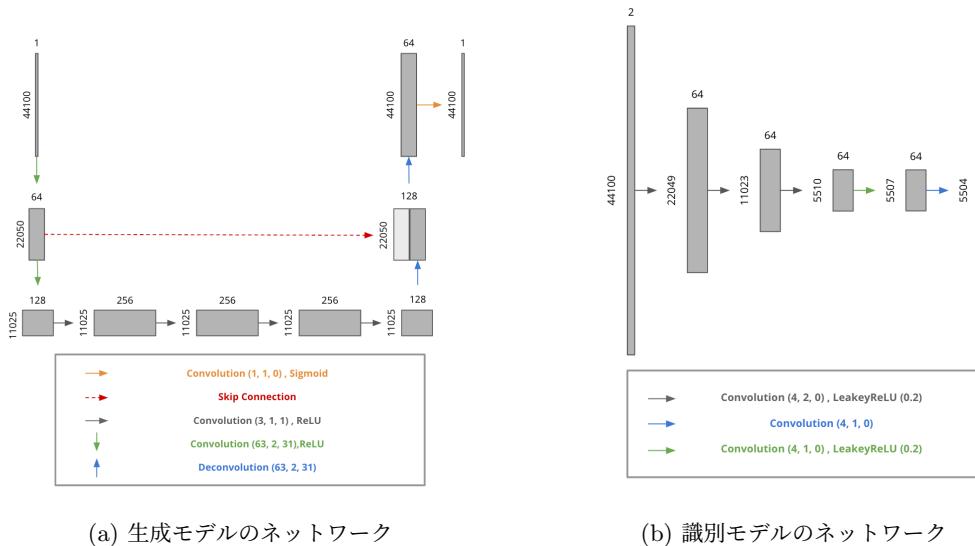


図 4.2: 本研究の生成モデルと識別モデル

第 5 章 実験

本章では、データセットと実験方法の説明を行った後、実験結果の考察を行う。

5.1 データセット

データセットとして 1 秒のギターとハープの音を 88 組用いた。また、この 88 音は音の大きさが等しいものと仮定した。そして、音の高さとしては A0~C8 の 88 音の半音を選んだ。これらは一般的な 88 鍵のピアノのそれぞれの鍵盤の音に対応する（図 5.1）。

5.1.1 音階表記

本実験では西洋音楽の 12 音階表記を音階表記に用いる。この表記では、 $C, C^\sharp, D, D^\sharp, E, F, F^\sharp, G, G^\sharp, A, A^\sharp, B$ の 12 段階の音の高さの集合をオクターブとして定める。そして、それぞれのオクターブに番号を振り、440 Hz の音を A4 と定めることで音の高さの絶対的な表記を可能にしている。

5.2 実験方法

本節では実験方法の説明を行う。

5.2.1 提案モデルの表現力の評価実験

提案モデルの表現力を評価する実験を行った。具体的には、用意した 88 音を学習データと評価データのいずれでも使用し、同じ高さの音の間での変換の実験を行った。

5.2.2 提案モデルの汎化能力の評価実験

任意の高さと大きさの音を学習データとして用意するのは不可能であるため、提案モデルの汎化能力を評価する必要がある。ここでは、88 音のうち $3/4$ を学習データ、 $1/4$ を評価データとする 4 分割交差検定により実験を行った。また、データセットの分割方法は付録 C に示す。

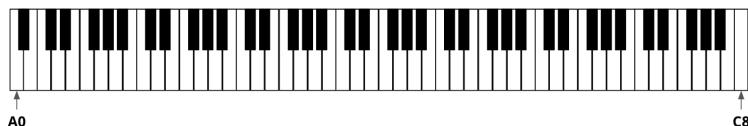


図 5.1: 88 鍵のピアノの鍵盤

5.2.3 データ拡張

各エポックの任意の学習データの振幅を無作為化することでデータ拡張を行った。具体的には、学習前に乱数のシードを固定した後に一様乱数により 0.3 ~ 1.0 の乱数を生成した。また、これにより 88 音と小さいサイズのデータセットへの過学習を防ぐことを期待した。

5.2.4 提案モデルの学習

モデルの学習時のパラメータの更新は 3.2.4.1 節にある勾配降下法を利用して行う。また、勾配降下法としては Adam [22] を用い、ハイパーパラメータは付録 B に示す。

5.2.5 評価方法

生成モデルの出力が音の高さ及び音の大きさを保持したまま変換先のハープの音の音色に変換できているかという観点から音波の観察と音の聴き取りにより評価を行った。また、音色の変換対象としてギターとハープを選んだ理由は、弦楽器という共通点を持ちながらも音波の観察と音の聴き取りの評価により十分に異なると判定できると考えたからである。

5.3 実験結果

実験結果の考察を本節では行う。また、実験結果に記載する波形の図は三つの波形を上から並べている。これらは上から順に、変換元のギターの波形、生成モデルの出力波形、変換先のハープの波形、である。そして、本節には一部の音波のみを記載し、付録 D に他の音波の波形を記載する。

5.3.1 概要：提案モデルの表現力の評価実験

提案モデルの表現力の評価実験を行ったところ、実験結果は二つに大別された。

5.3.1.1 ハープの音を表現できた場合

88 音のうち 86 音はハープの音を表現することができた。(図 5.2)。また、特に C4 から D5♯ は目標のハープの音に極めて近い音を生成することができた(図 5.3)。これらの音は上音が少ないので、他の音と比べて表現が容易であったと推察される。

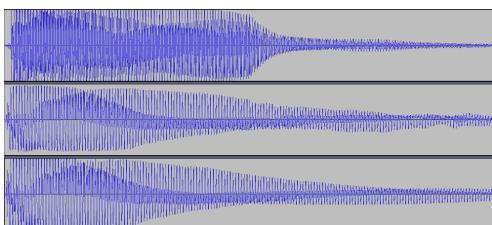


図 5.2: F3 の 0.000 秒から 1.000 秒まで

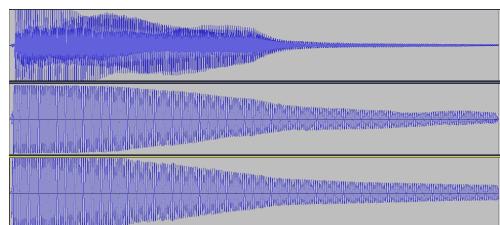


図 5.3: C4 の 0.000 秒から 1.000 秒まで

5.3.1.2 ハープの音を表現できなかった場合

D7♯とE7の2音は音の高さは維持できたものの、ハープの音を十分に表現できなかった（図5.4と図5.5）。いずれの音についてもハープの音波の振動が安定しておらず、この不安定さを表現することはできなかった。また、他の高音においても振動が不安定なものが見られたため、高音において安定したデータセットをハープで生成するのが難しいと推察される。

5.3.2 概要：提案モデルの汎化能力の評価実験

提案モデルの汎化能力の評価実験を行ったところ、実験結果は三つに大別された。

5.3.2.1 ハープの音を表現できた場合

D4,D4♯,G4,F5,F5♯の5音はハープの音を表現することができた（図5.6）。これらの高さの音では、提案モデルの表現力の評価実験の際にも特に綺麗なハープの音を表現できており、上音が少ないほど表現が容易であるという推察を示していると思われる。

5.3.2.2 ハープの音を表現できず音の高さも維持できなかった場合

C7,D7♯,E7,F7♯,G7,G7♯,A7,B7,C8の9音は音の高さも維持することができず、騒音が生成された（図5.7）。これらの高さの音では提案モデルの表現力の評価実験の際にもハープの音を表現できておらず、やはり高音域における安定したデータセットの作成は難しいと考えられる。また、高音域では周波数が高くノイズの影響が大きく出たとも考えられる。

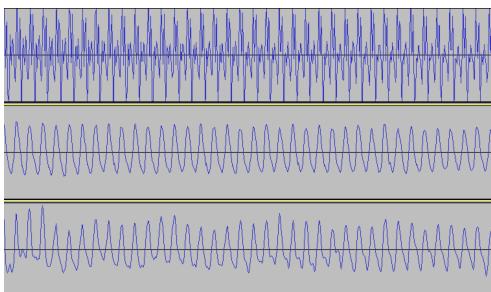


図5.4: D7♯の0.055秒から0.070秒まで

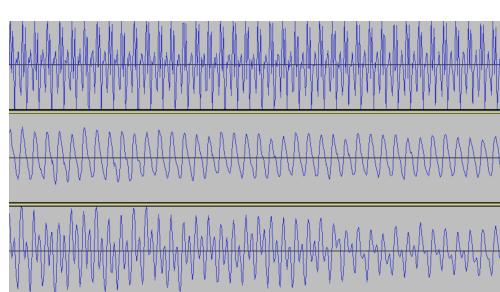


図5.5: E7の0.055秒から0.070秒まで

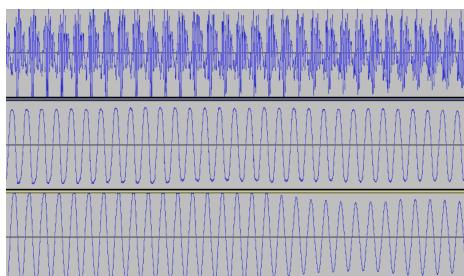


図5.6: D4♯の0.100秒から0.200秒まで

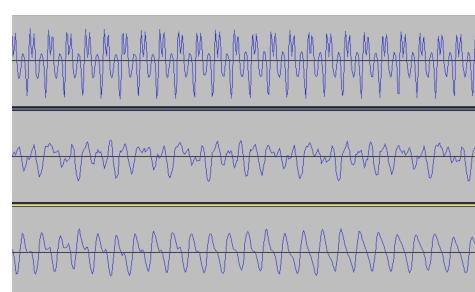


図5.7: D7♯の0.1700秒から0.110秒まで

5.3.2.3 ハープの音を表現できず音の高さを維持できた場合

14 音を除く 74 音については音の高さは維持できているもののハープの音を表現することができなかった（図 5.8、図 5.9）。これらの音では、音波が安定した振動をせず上音の成分がハープよりも多い音波が多く観測された。また、これらの高さの音は提案モデルの表現力の評価実験の際には表現することができていたため、提案モデルの汎化能力の低さが明らかになった。

5.3.3 課題

提案モデルの表現力の評価実験の実験結果から四つの課題が明らかになった。

5.3.3.1 音の減衰の表現

振動の減衰を表現できていない音がいくつかあった。これらの音のうち、E2 以下の低音域ではハープとは全く異なる波形で減衰し（図 5.10）、A6 以上の高音域ではほとんど振動が見られなかった（図 5.11）。また、提案モデルでは表現力が足りず微小な振動の学習が難しいためであると考えられる。

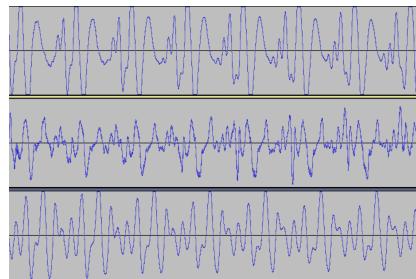


図 5.8: D1 の 0.300 秒から 0.500 秒まで

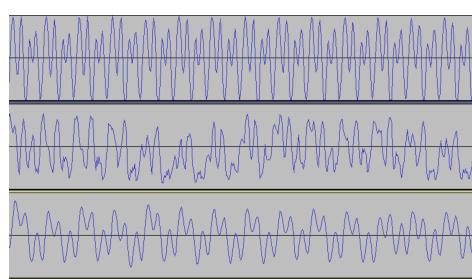


図 5.9: F6 の 0.070 秒から 0.080 秒まで

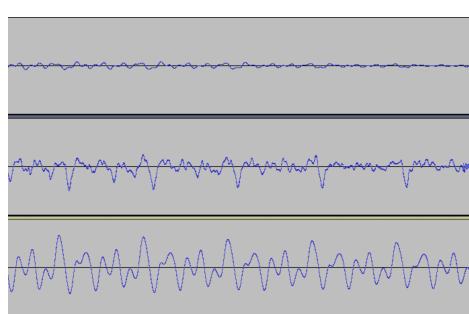


図 5.10: A0 の 0.800 秒から 1.000 秒まで

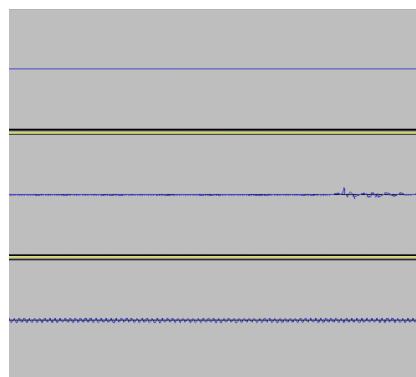


図 5.11: B7 の 0.980 秒から 1.000 秒まで

5.3.3.2 音の大きさの維持

部分的に大きさを維持できていない音がいくつかあった。図 5.12 では、波形の前半において生成波形の振幅が小さいことがわかる。原因は振幅をランダムに小さくしたことにあると考えられ、学習の初段階では振幅を固定しておくことや振幅の大きさを別で調整するなどの工夫が必要であると考えられる。

5.3.3.3 音波の滑らかさの表現

ノイズまじりの音がいくつかあった。これらの音を詳細に観察すると、生成された音波では滑らかさを表現できていないことがわかった(図 5.13)。音波を滑らかにするような加工を生成後に加えるなどの工夫が必要であると考えられる。

5.3.3.4 データセットの不安定さ

ここまで三つは提案モデルの改善により解消されると考えられるが、問題のあるデータセットも一部に存在した。一つの問題点は高音において振動が不安定になることであり、ここまで述べた。もう一つの問題点は音の鳴り出しでの振動が不安定であるという点である。具体的には、図 5.14 のようにギターの音の鳴り出しの遅延の方がハープより大きい場合や図 5.15 のように周期的な音になるまでに遅延があるために不規則な振動となる場合などがあった。

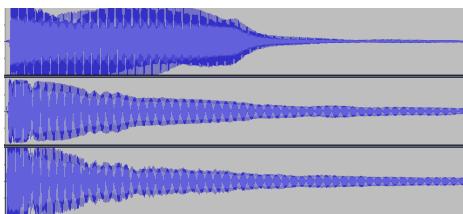


図 5.12: C5 の 0.000 秒から 1.000 秒まで

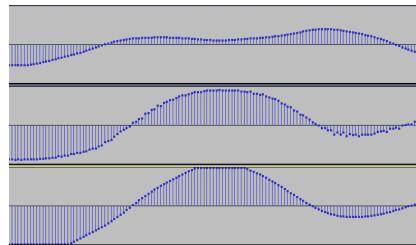


図 5.13: D2♯ の 0.100 秒から 0.103 秒まで

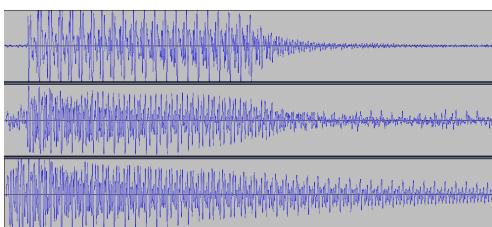


図 5.14: F1♯ の 0.000 秒から 1.000 秒まで

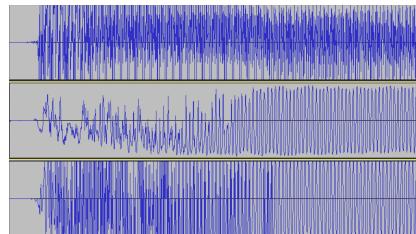


図 5.15: A7 の 0.000 秒から 0.030 秒まで

第6章　まとめ

本章では、実験結果の考察と展望についてまとめる。

6.1 実験結果のまとめ

提案手法の目標は音の高さと大きさを維持したままでギターの単音からハープの単音への音色の変換を行うことである。実験の結果、上記の目標に足る表現力を持つ提案モデルを作成できたが、汎化能力が十分ではないことも明らかになった。

また、汎化能力の低い理由としては主に二点あると考えられる。一点目は、Pix2pix [1] で用いられる Dropout 層を用いていない点である。二点目は、データ拡張の工夫が振幅の無作為化のみである点であり、時間方向にずらすなどの工夫もあると考えられる。

さらに、本研究では波形の観察を中心とした考察を行ったが、本研究の考察を明確化するためにはより定量的な判定を行う必要があると考えられる。具体的には、音の高さの維持、音の大きさの維持、音色の変換、という三点での定量的な判定を行う必要がある。

6.2 展望

本研究の実験が成功した場合、以下の三点を解決することで音楽の変換に適用できると考えられる。

6.2.1 楽器の重ね合わせ

音楽はそれぞれの楽器から出力される音の重ね合わせになっている。楽器ごとに音色は異なるため、音色変換を行うには楽器ごとの音波に分解すること（音源分離）が必要である。また、楽曲の作成時に楽器ごとに分離したデータ（パラデータ）の公開が一般的になれば音源分離の必要はなくなる。

6.2.2 音の重ね合わせ

ある音が単音の重ね合わせである場合は単音ごとに分離をして音色変換を行うことが必要である。また、本研究ではデータセットとして単音のみを使用するが、重音もデータセットに加えることで音色変換が可能であると考えられる。しかし、この際にデータセットが膨大な量になる可能性があり、追加するデータセットの工夫が必要である。

6.2.3 音の繋ぎ方

上記の二つが可能である時、時間方向の音の繋ぎ方を工夫する必要がある。研究前は単位時間ごとに区切るのみで可能であると考えたが、この場合はリズムの変更などで単位時間の基準が変わった際に対応することができない。しかし、この課題は二次元の表現を用いることで解決することができると考えられる。

謝辞

本研究を進める際にご指導を賜った指導教官の金子知適准教授に厚く感謝いたします。また、中屋敷太一さんをはじめとする金子研の構成員の方々には多くの助言をいただきました。ありがとうございました。

参考文献

- [1] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1125–1134, 2017.
- [2] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.
- [3] Keunwoo Choi, György Fazekas, Kyunghyun Cho, and Mark Sandler. A tutorial on deep learning for music information retrieval. *arXiv preprint arXiv:1709.04396*, 2017.
- [4] Sicong Huang, Qiyang Li, Cem Anil, Xuchan Bao, Sageev Oore, and Roger B Grosse. Timbretron: A wavenet (cyclegan (cqt (audio))) pipeline for musical timbre transfer. *arXiv preprint arXiv:1811.09620*, 2018.
- [5] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music. *arXiv preprint arXiv:2005.00341*, 2020.
- [6] Stanley Smith Stevens, John Volkmann, and Edwin Broomell Newman. A scale for the measurement of the psychological magnitude pitch. *The journal of the acoustical society of america*, Vol. 8, No. 3, pp. 185–190, 1937.
- [7] Stanley S Stevens and John Volkmann. The relation of pitch to frequency: A revised scale. *The American Journal of Psychology*, Vol. 53, No. 3, pp. 329–353, 1940.
- [8] Judith C Brown. Calculation of a constant q spectral transform. *The Journal of the Acoustical Society of America*, Vol. 89, No. 1, pp. 425–434, 1991.
- [9] Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Mohammad Norouzi, Douglas Eck, and Karen Simonyan. Neural audio synthesis of musical notes with wavenet autoencoders. In *International Conference on Machine Learning*, pp. 1068–1077. PMLR, 2017.
- [10] Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts. Gansynth: Adversarial neural audio synthesis. *arXiv preprint arXiv:1902.08710*, 2019.
- [11] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.
- [12] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
- [13] James J Callahan. *Advanced calculus: a geometric view*. Springer Science & Business Media, 2010.
- [14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, Vol. 25, pp. 1097–1105, 2012.
- [15] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Liyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. Recent advances in convolutional neural networks. *Pattern Recognition*, Vol. 77, pp. 354–377, 2018.

- [16] Kunihiko Fukushima and Sei Miyake. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition. pp. 193–202. Springer, 1980.
- [17] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.
- [18] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.
- [19] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pp. 1520–1528, 2015.
- [20] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.
- [21] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, Vol. 15, No. 1, pp. 1929–1958, 2014.
- [22] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

付録 A Adam

Adam は勾配降下法の中で最も用いられるアルゴリズムであり、図 A.1 に示す手順にしたがってパラメータの更新を行う。また、Adam については [22] に詳しい。

```
HyperParameter :  $\beta_1, \beta_2 \in (0, 1]$ ,  $\eta, \epsilon$ 
Objective Function :  $f$ 
Parameter :  $\theta$ 

Initialization:
   $t \leftarrow 0$                                      /* Initialize timestep */
   $\theta \leftarrow \theta_0$                                 /* Initialize Parameter */
   $m_1 \leftarrow 0$                                      /* Initialize 1st moment */
   $m_2 \leftarrow 0$                                      /* Initialize 2nd moment */

end

Procedure:
  while  $\theta$  not converged do
     $t \leftarrow t + 1$                                      /* Update timestep */
     $g \leftarrow \nabla_{\theta} f(\theta)$                       /* Compute gradient of  $f(\theta)$  */
     $m_1 \leftarrow \beta_1 \cdot m_1 + (1 - \beta_1) \cdot g$       /* Update biased 1st moment */
     $m_2 \leftarrow \beta_2 \cdot m_2 + (1 - \beta_2) \cdot (g \odot g)$   /* Update biased 2nd moment */
     $\hat{m}_1 \leftarrow m_1 / (1 - \beta_1^t)$                 /* Compute bias-corrected 1st moment */
     $\hat{m}_2 \leftarrow m_2 / (1 - \beta_2^t)$                 /* Compute bias-corrected 2nd moment */
     $\theta \leftarrow \theta - \eta \cdot \hat{m}_1 / (\sqrt{\hat{m}_2} + \epsilon)$  /* Update Parameter */

  end
  return  $\theta$ 
end
```

図 A.1: Adam の疑似コード

付録 B 学習時のパラメータ

表 B.1 に提案モデルの学習時のパラメータの値を示し、表 B.2 に Adam のハイパーパラメータの値を示す。

パラメータ	値
バッチサイズ	1
エポック数	1000

表 B.1

パラメータ	値
β_1	0.5
β_2	0.999
η	0.0002
ϵ	10^{-8}

表 B.2

付録 C データセットの分割

22 音ずつの 4 つのサブセットにデータセットを分割した (図 C.1)。また、データセットの 4 分割は、88 音をシャッフルして配列に格納した後に 22 音ずつ順に選ぶことで実装した。

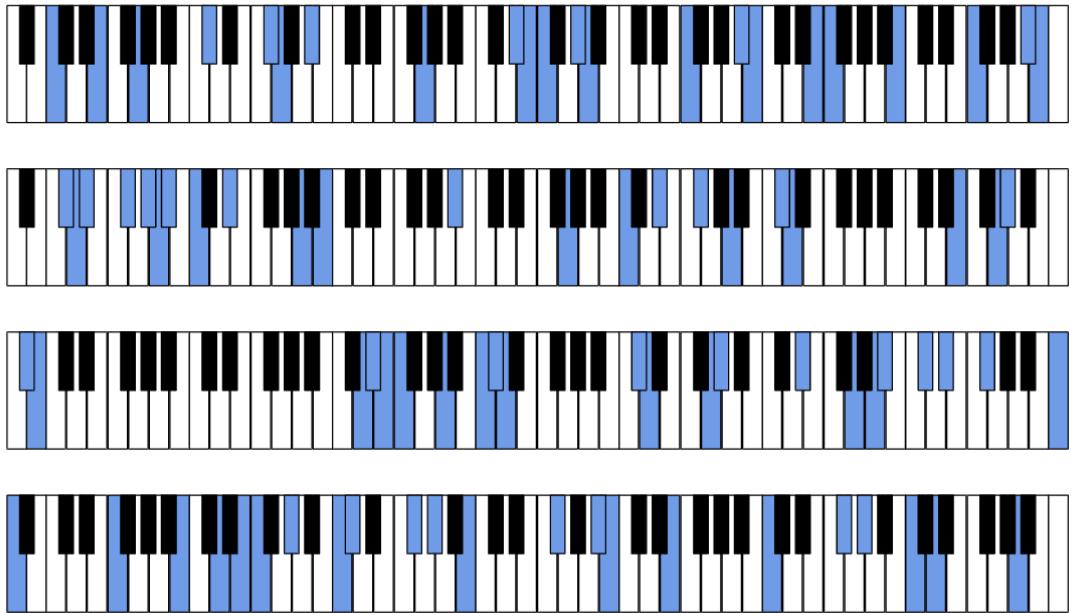


図 C.1: データセットのサブセット

付録 D 実験結果

5.3 節に載せることのできなかった波形の図を本章に載せる。また、波形の記載方法は 5.3 節と同様であるが、簡略化のため、キャプションには音階の表記のみ載せている。

D.1 提案モデルの表現力の評価実験

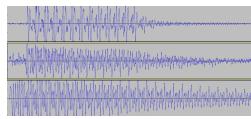


図 D.1: A0

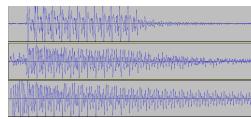


図 D.2: A0♯

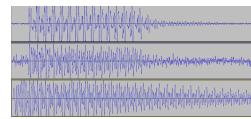


図 D.3: B0

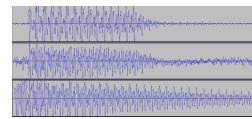


図 D.4: C1

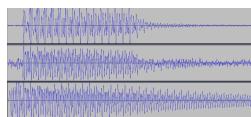


図 D.5: C1♯

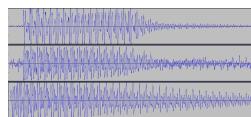


図 D.6: D1

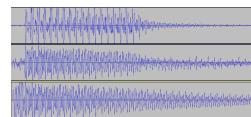


図 D.7: D1♯

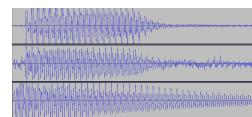


図 D.8: E1

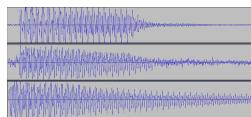


図 D.9: F1

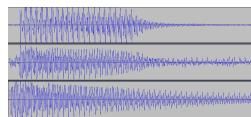


図 D.10: F1♯

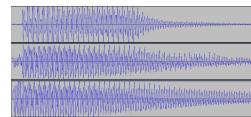


図 D.11: G1

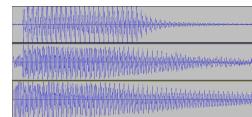


図 D.12: G1♯

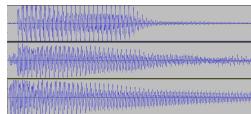


図 D.13: A1

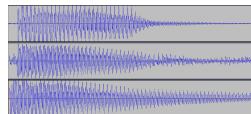


図 D.14: A1♯

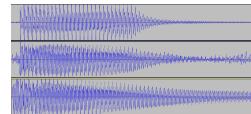


図 D.15: B1

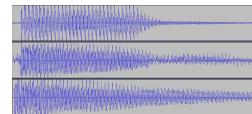


図 D.16: C2

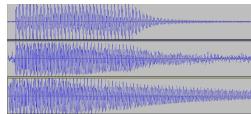


図 D.17: C2♯

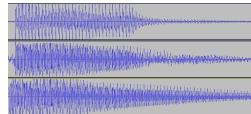


図 D.18: D2

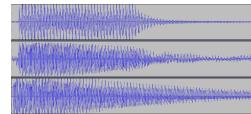


図 D.19: D2♯

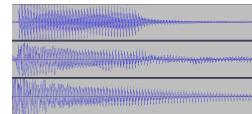


図 D.20: E2

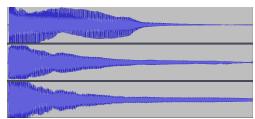


図 D.57: F5

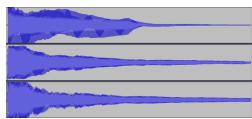


図 D.58: F5♯

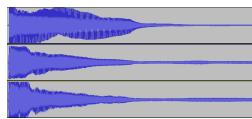


図 D.59: G5

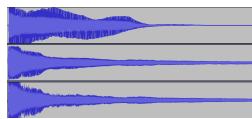


図 D.60: G5♯

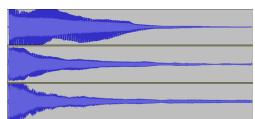


図 D.61: A5

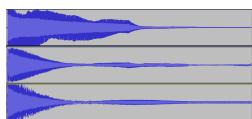


図 D.62: A5♯



図 D.63: B5

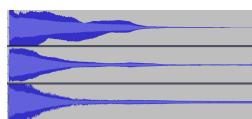


図 D.64: C6

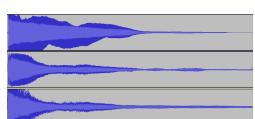


図 D.65: C6♯

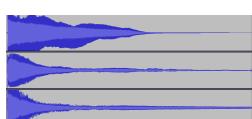


図 D.66: D6



図 D.67: D6♯



図 D.68: E6

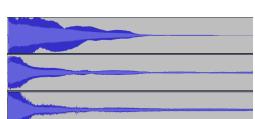


図 D.69: F6

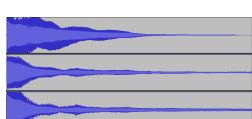


図 D.70: F6♯

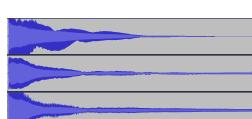


図 D.71: G6

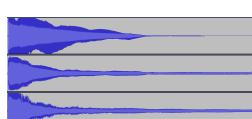


図 D.72: G6♯

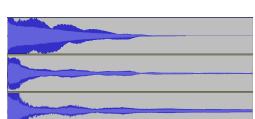


図 D.73: A6

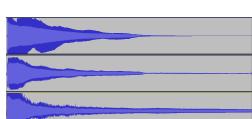


図 D.74: A6♯

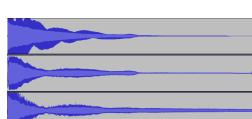


図 D.75: B6

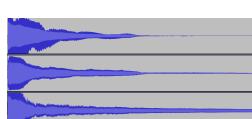


図 D.76: C7

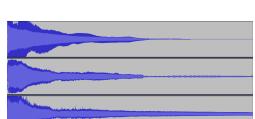


図 D.77: C7♯

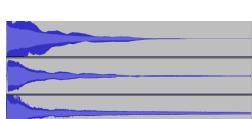


図 D.78: D7



図 D.79: D7♯

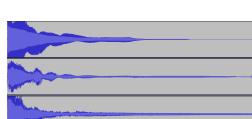


図 D.80: E7

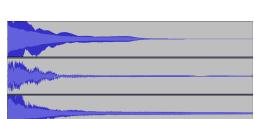


図 D.81: F7



図 D.82: F7♯

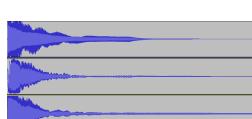


図 D.83: G7

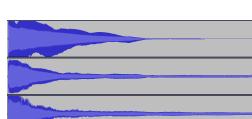


図 D.84: G7♯



図 D.85: A7



図 D.86: A7♯



図 D.87: B7



図 D.88: C8

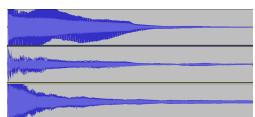


図 D.149: A5

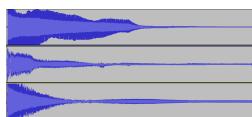


図 D.150: A5♯

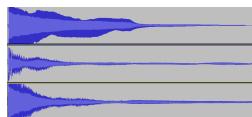


図 D.151: B5

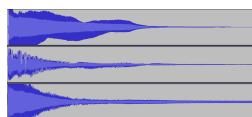


図 D.152: C6

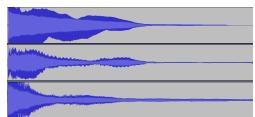


図 D.153: C6♯

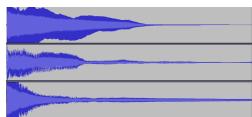


図 D.154: D6

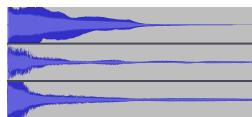


図 D.155: D6♯

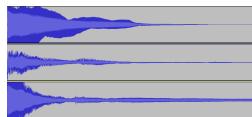


図 D.156: E6

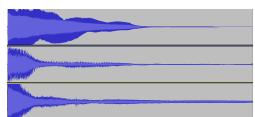


図 D.157: F6

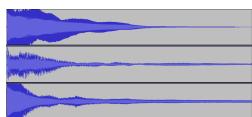


図 D.158: F6♯

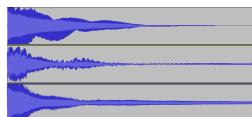


図 D.159: G6

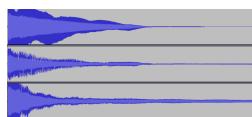


図 D.160: G6♯

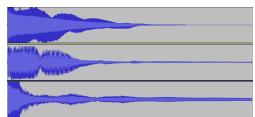


図 D.161: A6

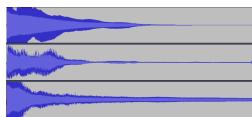


図 D.162: A6♯

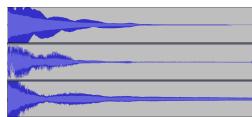


図 D.163: B6

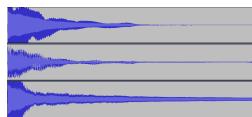


図 D.164: C7

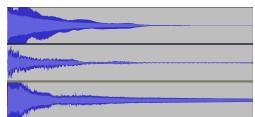


図 D.165: C7♯

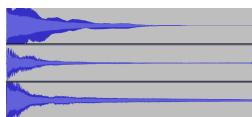


図 D.166: D7



図 D.167: D7♯



図 D.168: E7



図 D.169: F7



図 D.170: F7♯



図 D.171: G7



図 D.172: G7♯



図 D.173: A7



図 D.174: A7♯



図 D.175: B7



図 D.176: C8