Vol.2018-MUS-120 No.12
2018/8/22

畳み込みニューラルネットワークと敵対的学習に基づくメロディ補完の検討

中村 光佑[1,a)]　能勢 隆[1]　千葉 祐弥[1]　伊藤 彰則[1]

概要：本稿では，部分的に欠損させたメロディを自然なかたちで補完する問題を扱う．メロディの補完は，既存のメロディの部分的な編集や 2 つのメロディの接続といった用途へ利用が期待される．欠損した情報を補完する技術としては，画像のある領域を欠損させ，その部分を自然に補完する画像補完が研究されており，近年ではニューラルネットワークによって高品質な画像補完を実現する手法が提案されている．そこで本研究では，ニューラルネットワークに基づく画像補完を応用したメロディの補間手法を検討する．具体的には，一定長のメロディを画像として表現し，これを用いて画像補完ネットワークを学習する．補完ネットワークは全層が畳み込み層で構成され，敵対的学習の枠組みで学習される．敵対的学習では，本物のメロディと補完されたメロディを見分ける識別ネットワークを同時に学習させ，補完ネットワークはこれをだませるように学習を行うことで，より自然な補完結果が得られると期待される．また，学習時の補助情報として，楽曲のコードの情報を入力することも検討する．

キーワード：メロディ補完，自動作曲，畳み込みニューラルネットワーク，敵対的学習

1. はじめに

映像作品やゲームの BGM，テーマソングとしての利用など，新規楽曲が必要されることは多いが，それを必要とする人が作曲経験者であるとは限らない．しかし，音楽経験や作曲の技術を持たない人が自身の好みや用途にあった楽曲を作るのは難しく，楽曲の作成を支援するシステムの需要がある．これを実現する技術の一つとして，自動作曲システムの研究がある．自動作曲システムは，ユーザの入力をもとに楽曲を自動で生成するシステムであり，これまで，確率モデルに基づいた Orpheus[1]，Long Short-Term Memory に基づいた Performance RNN[2]，Deep Convolutional Generative Adversarial Network に基づいた MidiNet[3] といった手法が提案されてきた．

これらの手法の多くは歌詞やジャンルといった情報，音の多さ等のパラメータやノイズベクトルなどの入力をもとに楽曲を生成する．一方で，作曲技術を持たない人が特定の既存楽曲に似た雰囲気の曲を作りたいとき，その楽曲をベースに曲を生成する手法が有用であると考えられる．そこで本研究では，部分的に欠損させた楽曲のメロディを自然な形で補完する，メロディ補完を検討する．メロディ補完は，メロディの欠損と補完により既存楽曲の編集を実現する．また，楽曲の先頭から逐次的に欠損と補完を繰り返すことで，既存楽曲をもとに，同じ構成やコード進行を持つ曲を新たに生成することが出来ると考えられる．佐々木らは，ニューラルネットワークを用いて 12 個の連続した音符のうち中央の 4 つの音符の音高を予測するメロディの補完ネットワークを提案した [4]．この手法では入力層の各ノードに音符の音高を与えるため，予測されるのは音高のみで，リズムは元のメロディのものしか使用できないという問題があった．

一方で，欠損した情報を補完する技術としては，画像の一部が削除されたり欠落した場合に，その部分を自然に補完する画像補完が研究されており，近年ではニューラルネットワークによって高品質な画像補完を実現する手法が提案されている．Iizuka らは，畳み込みニューラルネットワーク（CNN: Convolutional Neural Network）による画像補完ネットワークと，画像全体の大域的な整合性，補完領域周辺の局所的な整合性を見る 2 つの識別ネットワークによる敵対的学習により非常に自然な画像補完が出来る手法を提案している [5]．画像表現されたメロディをこの手法により補完することで，音高だけでなく，リズムも考慮したメロディの補完を行うことができると考えられる．

本研究では，音高だけでなくリズムも考慮したメロディ補完を実現する為に，文献 [5] を応用し，メロディを画像

[1] 東北大学
[a)] kosuke.nakamura@spcom.ecei.tohoku.ac.jp

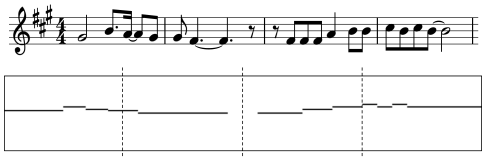


図 1 メロディ画像の例．上の楽譜を変換したものが下の画像で，点線が小節線に対応する．見やすさのため白黒を反転してある

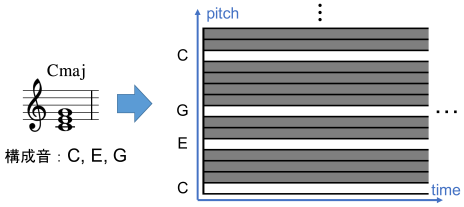


図 2 C メジャーコードの画像表現

として表現することで，画像補完の枠組みによるメロディ補完を検討する．

以降では，まず第 2 節で，メロディとコードを画像として表現する方法とそれを用いて作成したデータセットについて説明する．次に第 3 節では補完を行うネットワークとその学習方法について述べる．第 4 節では学習したネットワークによりメロディ補完を行い，その結果を考察する．

2. メロディとコードの画像表現

本節では，画像補完を応用したメロディ補完に必要な，メロディとコードの画像表現について説明する．また学習に使用したデータセットについても説明する．

2.1 メロディの画像表現

画像補完の枠組みを用いてメロディ補完を実現するために，メロディを一枚の画像として表現する必要がある．本研究では，楽曲中からメロディを一定の長さで切り出し，これらを画像に変換したものを補完する．

メロディの画像への変換は，以下のように行った．画像の縦方向を音高に，横方向を拍・時間に対応させ，楽譜上で音が存在する位置に対応したピクセルは白で，それ以外のピクセルは黒で塗りつぶす．これにより，メロディをピアノロール状の白黒の画像として表すことができる．以下ではこの画像をメロディ画像と呼ぶ．ネットワークに入力する際には，白い部分が 1，黒い部分が 0 の値を持つ行列として扱われる．ただし，本稿では見やすさの為に白黒を反転させた画像を載せる．縦方向の 1 ピクセルは半音に相当し，画像の最下行を最低音として上のピクセルほど高い音に対応する．実験に用いたメロディの最低音と最高音が含まれる C2 から B6 までの 5 オクターブを表現出来るように，画像の高さは 60（5 オクターブ ×12 半音）ピクセルとした．1 枚の画像として表されるメロディの長さは 4 小節とし，4 分音符 1 つの長さを 24 ピクセルとした．これは，32 分音符と 3 連符の長さを表現できるように決定した．対象とする曲を 4/4 拍子に限定することで，画像の幅を $24 \times 4 \times 4 = 384$ とした．この方法で変換した画像と元のメロディの例を**図 1** に示す．

2.2 コードの画像表現

メロディ補完を行う際，対応する小節のコード進行を入力に加えることで，補完されるメロディの音高を決定する際の補助情報として機能すると考えた．メロディ画像と同じサイズの画像として表現することで，メロディとコードの情報を画像のチャンネル方向に重ねるだけでネットワークに同時に入力することができる．

コードの画像への変換は，基本的には前節で述べたメロディの画像化と同じルールのもとで行う．各拍において，コードの構成音に対応する行を全てのオクターブについて白く塗る．**図 2** は C メジャーコードを表す画像の一部を示している．C メジャーコードの構成音は C，E，G の 3 つである．図中の矩形の最下部が C なので，この行は白く塗られる．1 行上がるごとに音高が半音ずつ上がっていき，E，G に対応する行も白く塗られる．下から 13 行目は 1 オクターブ上の C であり，ここも白く塗る．このようにして画像の上端まで対応する行を白く塗ることで，コードを表現することができる．この手法によりコードを表現することで，ネットワークがコード構成音とメロディの間の関係をより適切に学習できると期待される．

2.3 データセット

ネットワークの学習用，評価用のデータセットを構築した．変換元となるメロディには，株式会社インターネット[*1]で取り扱われている，日本のポップスを中心とした 288 曲のボーカル楽曲のデータを用いた．メロディは全て単旋律である．データはテキストイベントとしてコード進行の情報を持つ MIDI データで，MuseScore2[*2]を用いて一度楽譜データである MusicXML 形式に変換した上で画像へ変換した．メロディの切り出しは，4 小節の長さの窓を 1 小節ずつシフトしながら行い，簡単化のため全休符が含まれるような 4 小節は切り出さずにとばした．これにより 23,392 枚のメロディ画像を得た．また，これらのメロディ画像に対応するコード画像も作成した．これらを学習データ 20,041 組（メロディ画像 1 枚と対応するコード画像 1 枚の組）と評価データ 3,351 組に分けた．データを分ける際には，学習データと評価データに同一の楽曲から得られたメロディ画像が含まれないように分割した．また，全てのメロディとコードを，元の曲が長調であればハ長調に，元の曲が短調であればイ短調に移調した．

[*1] https://www3.ssw.co.jp/dl_data
[*2] https://musescore.org

3. メロディ補完ネットワーク

ここでは，本研究でメロディ補完に用いるネットワークの構造と，その学習方法について述べる．

3.1 構造

ネットワークには，文献 [5] の画像補完ネットワークと同様の構造を用いた．**図 3** に，ネットワークの構造を表す概略図を示す．補完ネットワークには，メロディ画像と，そのメロディに対応したコード画像，及び補完対象領域を示すマスク画像（0:補完対象領域外，1:補完対象領域）が入力される．メロディ画像は補完対象領域をデータセット全体のピクセルの値の平均値で塗りつぶされる．補完ネットワークは，通常の convolution 層と dilated convolution 層および deconvolution 層からなる．dilated convolution 層はフィルタを適用するピクセルを dilation の値ごとに選択することでより広範囲の特徴を抽出することができる畳み込み層 [6] である．deconvolution 層はフィルタのシフト幅を 1 未満の分数とすることで通常の convolution とは逆に特徴マップのサイズを大きくするアップサンプリングを行う [7]．出力層以外の全ての層の後段に活性化関数として Rectified Linear Unit（ReLU）関数を，出力層の活性化関数には sigmoid 関数を用いた．補完ネットワークは補完結果として 1 チャンネルの画像を出力する．この画像の補完対象でない領域を入力の画像に置き換え，これを最終的な補完結果とする．

識別ネットワークにはこの補完された画像かマスク前のメロディ画像，それに加えて対応するコード画像が入力される．識別ネットワークは大域識別ネットワークと局所識別ネットワークの 2 つのネットワークからなり，それぞれ画像全体の整合性，補完対象領域周辺の整合性を評価する．大域識別ネットワークには画像全体を入力し，局所識別ネットワークには補完対象となった領域のみを切り取って入力する．コードの情報もこの領域に合わせて切り取って入力される．本物の画像を局所識別ネットワークに入力する場合は，同様のサイズの領域をランダムに切り取って入力する．2 つのネットワークの出力は全結合層によってまとめられ，sigmoid 関数を通して入力された画像が本物の画像である確率を出力する．

各層の出力チャンネル数，フィルタサイズ，ストライド幅は文献 [5] にならい**表 1**，**表 2** に示すように設定した．2 つの識別ネットワークは，表中の全結合層の出力を連結した後にさらに全結合層によって 1 つのスカラー値を出力する．

3.2 学習

ネットワークの学習は，敵対的学習の枠組みを用いて行われる．文献 [5] において，補完ネットワークの学習は，2 つの損失関数を組み合わせた損失を最小化するように行われる．一つは補完結果と入力画像の，補完対象領域に重み付けされた Measn Squared Error（MSE）[8] で，下式で表される．

$$L(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{m_c}) = ||\boldsymbol{m_c} \odot (C(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{m_c}) - \boldsymbol{x})||^2 \qquad (1)$$

ここで，$C(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{m_c})$ は入力メロディ画像 $\boldsymbol{x}$ とコード画像 $\boldsymbol{y}$，これらと同サイズのマスク画像 $\boldsymbol{m_c}$ を補完ネットワークに入力した時の出力である．メロディ画像 $\boldsymbol{x}$ の座標 (i, j) $(0 \leq i \leq 383, 0 \leq j \leq 59)$ の要素 $x_{ij} \in \{0, 1\}$ は，メロディが座標 (i, j) に対応する拍・音高で鳴っていれば $x_{ij} = 1$，鳴っていなければ $x_{ij} = 0$ の値をとる．$\boldsymbol{y}$ はその拍で鳴っ

表 1 補完ネットワークの構造．conv. は通常の畳み込み層を，dilated は dilated convolution 層を，deconv. は deconvolution 層を表す

Type	Filter	Dilation	Stride	Outputs
conv.	5×5	1	1×1	64
conv.	3×3	1	2×2	128
conv.	3×3	1	1×1	128
conv.	3×3	1	2×2	256
conv.	3×3	1	1×1	256
conv.	3×3	1	1×1	256
dilated	3×3	2	1×1	256
dilated	3×3	4	1×1	256
dilated	3×3	8	1×1	256
dilated	3×3	16	1×1	256
conv.	3×3	1	1×1	256
conv.	3×3	1	1×1	256
deconv.	4×4	1	$1/2 \times 1/2$	128
conv.	3×3	1	1×1	128
deconv.	3×3	1	$1/2 \times 1/2$	64
conv.	3×3	1	1×1	32
output	3×3	1	1×1	1

表 2 識別ネットワークの構造．conv. は通常の畳み込み層を，FC は全結合層を示す．

大域識別ネットワーク

Type	Filter	Stride	Outputs
conv.	5×5	2×2	64
conv.	5×5	2×2	128
conv.	5×5	2×2	128
conv.	5×5	2×2	256
conv.	5×5	2×2	512
conv.	5×5	2×2	512
conv.	5×5	2×2	512
FC	–	–	1024

局所識別ネットワーク

Type	Filter	Stride	Outputs
conv.	5×5	2×2	64
conv.	5×5	2×2	128
conv.	5×5	2×2	128
conv.	5×5	2×2	256
conv.	5×5	2×2	512
conv.	5×5	2×2	512
FC	–	–	1024

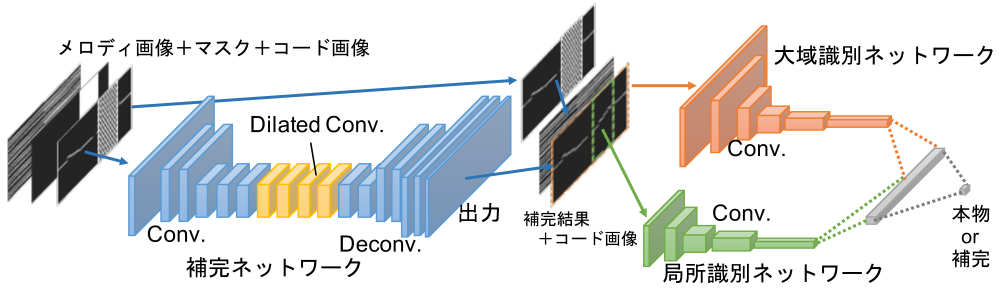


図 3　ネットワークの構造

ているコードの構成音にあたる音高に対応する座標の要素が 1，それ以外が 0 となる．マスク画像 $\boldsymbol{m_c}$ は補完対象領域内の要素は 1 を，それ以外は 0 の値をとる．また $\odot$ は要素単位での積を表し，$||\cdot||$ はユークリッドノルムを表す．

もう一つは，下式で表される GAN loss[9] である．

$$\begin{aligned}&\log D(\boldsymbol{x},\boldsymbol{y},\boldsymbol{m_d})\\&\quad+\log\left(1-D\left(C\left(\boldsymbol{x},\boldsymbol{y},\boldsymbol{m_c}\right),\boldsymbol{y},\boldsymbol{m_c}\right)\right)\end{aligned}\tag{2}$$

ここで，$\boldsymbol{m_d}$ は補完対象領域が $\boldsymbol{m_c}$ と同じ大きさのマスク画像でマスクの位置はランダムに決定される．$D(\boldsymbol{x},\boldsymbol{y},\boldsymbol{m_d})$ は識別ネットワークの出力を表す．この値が大きいほど識別ネットワークは画像を正しく識別できていることを示す．補完ネットワークはこの値を最小化するように，識別ネットワークは最大化するように学習される．

2 つの損失関数を組み合わせた損失関数は，重み係数 α を用いて，

$$\begin{aligned}&L(\boldsymbol{x},\boldsymbol{y},\boldsymbol{m_c})+\log D(\boldsymbol{x},\boldsymbol{y},\boldsymbol{m_d})\\&\quad+\alpha\log\left(1-D\left(C\left(\boldsymbol{x},\boldsymbol{y},\boldsymbol{m_c}\right),\boldsymbol{y},\boldsymbol{m_c}\right)\right)\end{aligned}\tag{3}$$

と書かれる．結局，補完ネットワークと識別ネットワークはそれぞれ，この関数を最小化，最大化するように最適化される．最適化の手法には ADADELTA[10] を用いる．

また，学習の安定化のため，敵対的学習を行う前に，まず T_C エポックだけ MSE のみを損失関数として補完ネットワークを単独で学習させる．次に T_D エポックだけこの補完ネットワークの出力と本物の画像を識別するように識別ネットワークを単独で学習させる．その後，式 (3) の損失関数を用いて敵対的学習を T エポック行う [5]．

さらに，学習時に識別ネットワークに与える識別結果の正解ラベルを偽物の画像に対して 0，本物の画像に対して $1-\beta$ $(0<\beta\ll 1)$ とすることで正則化の効果があるとされる One-sided label smoothing[11][12] を用いた．これにより，識別ネットワークが最大化する式 (2) は以下のようになる．

$$\begin{aligned}&(1-\beta)\log D(\boldsymbol{x},\boldsymbol{y},\boldsymbol{m_d})+\beta\log\left(1-D\left(\boldsymbol{x},\boldsymbol{y},\boldsymbol{m_d}\right)\right)\\&\quad+\log\left(1-D\left(C\left(\boldsymbol{x},\boldsymbol{y},\boldsymbol{m_c}\right),\boldsymbol{y},\boldsymbol{m_c}\right)\right)\end{aligned}\tag{4}$$

表 3　学習条件

ミニバッチサイズ	20	エポック数 (T_C)	40
α	0.0004	エポック数 (T_D)	20
β	0.1	エポック数 (T)	40

4. 実験

本節では，2 節で説明したデータセットを用いてネットワークを学習し，メロディ補完を行った結果とそれに対する考察を述べる．

4.1 実験条件

学習は次のような条件で行った．メロディ画像は入力時に 2 小節目か 3 小節目をランダムに欠損させる．欠損させる領域のサイズは 1 小節分にあたる 60×96 となる．補完ネットワークの出力を局所識別ネットワークに入力する際には，この補完対象領域にあたる部分を切り取って入力する．学習は一定の数の入力の後にパラメータを更新するミニバッチ学習によって行った．**表 3** にミニバッチサイズ，エポック数，重み係数 α 及び One-sided label smoothing の式 (4) の β の値を示す．また，補助情報としてのコード画像の有効性の有無を確認するために，コード画像を用いる場合と用いない場合の各条件で学習・補完を行った．

4.2 補完結果

4.2.1 定性的評価

図 4 に，クローズドな入力に対する各条件での補完結果を 2 例示す．図中の w/o chord, w/ chord はそれぞれコードを入力として与えない場合と与える場合を表す．また MSE，GAN はそれぞれ MSE のみを損失関数として T_C エポックだけ学習した後の補完ネットワークと，敵対的学習後の補完ネットワークの補完結果を表す．図 4 の MSE のみで学習したモデルの補完結果を見ると，コードの有無に関わらず，入力画像に似た結果を出力できていることが確認できる．コードを与えた場合の敵対的学習後のモデルの補完結果も，入力されたメロディ画像に近いメロディを出力できているが，図 4 中の 2 つの例も含めた多くの結果において，補完領域全体にノイズのようなものが現れた．ま

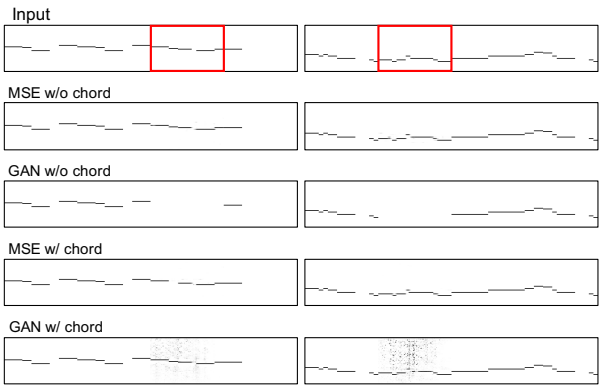


図 4 クローズドな入力に対する補完結果の例．見やすさのため白黒を反転してある．Input の赤枠内が補完対象領域．列がそれぞれ同一の入力に対する出力を示す

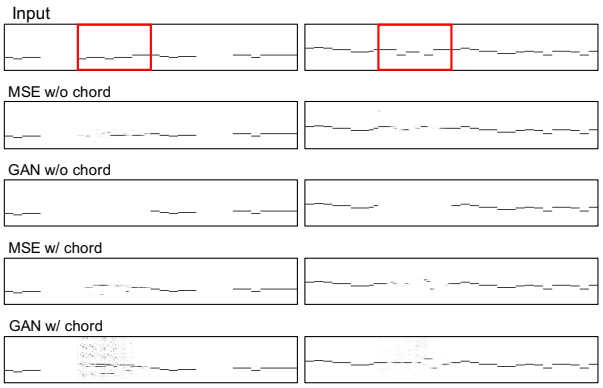


図 5 オープンな入力に対する補完結果の例．見やすさのため白黒を反転してある．Input の赤枠内が補完対象領域．列がそれぞれ同一の入力に対する出力を示す

た，コードを与えない場合，音を一つも出力しないような補完結果となる場合がほとんどであった．こうしたことから，ネットワークは 4 小節のメロディの構造を把握することはできているが，敵対的学習が上手く機能していないと考えられ，重み係数 α や補完ネットワークと識別ネットワークのチャンネル数などのパラメータの検討が必要だと考えられる．

図 5 に，評価データを入力したときの，各条件での補完結果を 2 例示す．補完結果を見ると，コードを用いない敵対的学習後のモデルを除き，いずれも線がかすれ，ちぎれがちな画像を出力しているが，音高に関して言えば，入力画像に比較的近い位置に線が出力されている．クローズドな入力に対しては正解に近い補完ができていたことから，ネットワークが過学習していると考えられる．したがって，ネットワークのパラメータ数を絞る等の方策により，ネットワークが汎化する学習条件を検討する必要があると考えられる．

4.2.2 定量的評価

前項では各条件において補完結果を観察し，定性的な評価を行ったが，正解データに非常に近い見た目の補完がなされた条件を除き，実際に出力されたメロディの妥当性を判断することは難しい．そこで，補完結果の妥当性を定量的に評価する指標の一つとして，出力されたメロディの音がコードの構成音に含まれている割合を計算し，各条件ごとに比較した．メロディに使用される音とコードの間には相互依存的な関係があると考えられ，メロディの音としてコードの構成音に含まれる音が使用されることも多い．したがってこの割合は，ネットワークがメロディに使用できる音高についてどの程度学習できたかを図る目安になると考えられる．

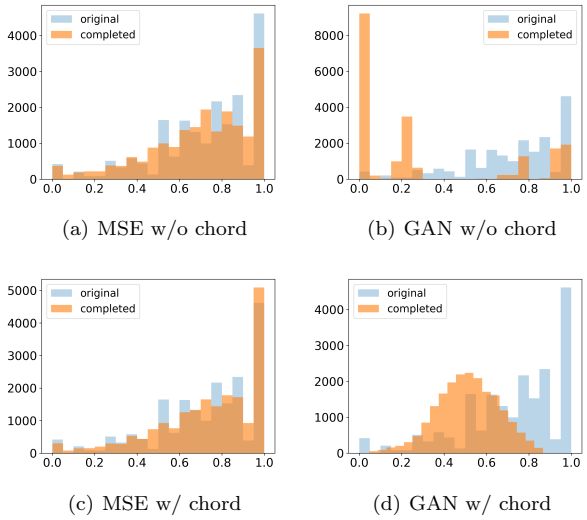
メロディのうちコードの構成音上に置かれたものの割合は，メロディ画像 $\boldsymbol{x}$，コード画像 $\boldsymbol{y}$，マスク画像 $\boldsymbol{m}$ に対して，下式で計算される．

$$R(\boldsymbol{x},\boldsymbol{y},\boldsymbol{m}) = \frac{\sum_{i,j} x_{ij} \times y_{ij} \times m_{ij}}{\sum_{i,j} x_{ij} \times m_{ij}} \qquad (5)$$

$R(\boldsymbol{x},\boldsymbol{y},\boldsymbol{m})$ は，大きいほど補完対象領域のメロディに使われている音がコードの構成音であることが多いことを示す．ネットワークがコードの情報を活かしてメロディの補完をできている場合，補完結果の画像に対するこの値が，本物の画像に対する値と近い分布になると考えられる．なお，補完ネットワークが出力する画像は，同一時刻に複数の音高に対応するピクセルに値を持つことがあるが，コード構成音上に音を出力しようとする度合いを評価する意味で，これらはそのままにして計算した．

図 6 に，学習データを入力したとき，補完結果のメロディがコード構成音に含まれている割合のヒストグラムを示す．横軸は $R(\boldsymbol{x},\boldsymbol{y},\boldsymbol{m})$，縦軸は度数を示す．各図中で青色で示されているのは，学習データ中の各メロディに対して，補完対象領域と同じ領域のメロディの音がコード構成音に乗っている割合のヒストグラムである．図 6 の (a) と (c) をみると，補完ネットワークのみでの学習後のモデルのヒストグラムは，コードの有無によらず本物の画像のものに近い．これは前項でクローズドな入力に対しては正確な画像が出力されていることを反映している．これに対して，(b) と (d) を見ると，敵対的学習後のモデルのヒストグラムは本物の画像のものと大きく異なる．特にコードを入力しない場合では，全体の半分近くが 0 から 0.05 という値をとっている．これは前項で確認されたように，この条件において，補完ネットワークが対象領域を全て休符として補完することが多いためである．

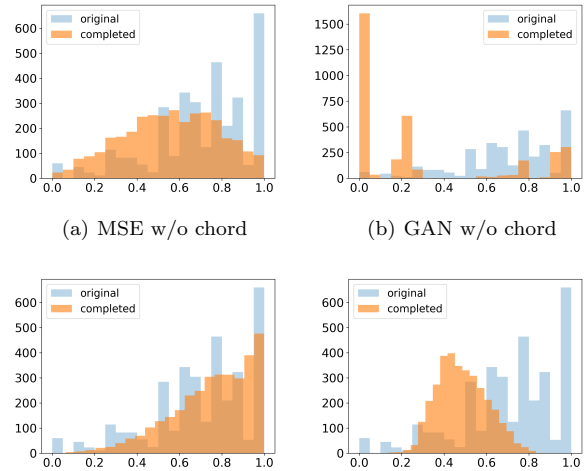
図 7 に，評価データを入力したとき，補完結果のメロディがコード構成音に含まれている割合のヒストグラムを示す．**図 7(a)** を見ると，補完結果のヒストグラムは評価データのヒストグラムと大きく異なる山を描いている．コードを付与して学習した (c) では，補完結果のヒストグラムと評価データのヒストグラム概形が近いものとなって



(a) MSE w/o chord　(b) GAN w/o chord

(c) MSE w/ chord　(d) GAN w/ chord

図 6 クローズドな入力に対する補完結果のメロディがコード構成音である割合のヒストグラム．横軸が $R(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{m})$，縦軸が度数を示す（completed:補完結果のヒストグラム，original:学習データのヒストグラム）



(a) MSE w/o chord　(b) GAN w/o chord

(c) MSE w/ chord　(d) GAN w/ chord

図 7 オープンな入力に対する補完結果のメロディがコード構成音である割合のヒストグラム．横軸が $R(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{m})$，縦軸が度数を示す（completed:補完結果のヒストグラム，original:評価データのヒストグラム）

いる．このことから，図 5 では，補完結果のメロディは短くちぎれがちで自然な見た目ではなかったが，音高に関して言えば比較的妥当な位置にメロディを出力しているといえる．**図 7(b)** は**図 6(b)** と同様，そもそも補完結果として出力する音が非常に少ないために，評価データに対するヒストグラムと大きく異なった形となっている．

コード画像の有無で比較すると，ほとんどの条件においてコードを与える場合の方が本物のデータのヒストグラムに近い形となることから，コードの情報を与えることの有効性を確認できたといえる．

5. まとめ

メロディとコードを画像として表現し，CNN と敵対的学習に基づく画像補完を適用することで欠損させたメロディを補完するシステムを提案した．補完結果から，ネットワークはクローズドな入力に対しては妥当な補完を行えることが確認できた．また，コードの情報を与えることの有効性を確認した．一方で，オープンな入力に対しては正しく補完を行えないこと，敵対的学習が上手く機能しておらず補完の品質が劣化してしまうことが確認された．

今後の課題としては，敵対的学習の安定化とネットワークの汎化を目指したハイパーパラメータの検討や，補完結果に多様性を持たせるための手法の検討などが挙げられる．

参考文献

[1] 深山　覚，中妻　啓，米林裕一郎，酒向慎司，西本卓也，小野順貴，嵯峨山茂樹：Orpheus: 歌詞の韻律に基づいた自動作曲システム，情報処理学会研究報告， Vol. 2008, No. 78 (2008-MUS-076), pp. 179–184 (2008).

[2] Simon, I. and Oore, S.: Performance RNN: Generating Music with Expressive Timing and Dynamics, `https://magenta.tensorflow.org/performance-rnn` (2017).

[3] Yang, L.-C., Chou, S.-Y. and Yang, Y.-H.: MidiNet: A Convolutional Generative Adversarial Network for Symbolic-domain Music Generation using 1D and 2D Conditions, *arXiv preprint arXiv:1703.10847* (2017).

[4] 佐々木真菜，坂井拓己，平井辰典：メロディの自動補間によるメロディ編集手法の検討，情報処理学会研究報告，Vol. 2018-SLP-122, No. 24, pp. 1–4 (2018).

[5] Iizuka, S., Simo-Serra, E. and Ishikawa, H.: Globally and Locally Consistent Image Completion, *ACM Transactions on Graphics (Proc. of SIGGRAPH 2017)*, Vol. 36, No. 4, pp. 107:1–107:14 (2017).

[6] Yu, F. and Koltun, V.: Multi-Scale Context Aggregation by Dilated Convolutions, *arXiv preprint arXiv:1511.07122* (2015).

[7] Long, J., Shelhamer, E. and Darrell, T.: Fully Convolutional Networks for Semantic Segmentation, *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431–3440 (2015).

[8] Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T. and Efros, A. A.: Context encoders: Feature learning by inpainting, *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2536–2544 (2016).

[9] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y.: Generative Adversarial Nets, *Advances in Neural Information Processing Systems 27*, Curran Associates, Inc., pp. 2672–2680 (2014).

[10] Zeiler, M. D.: ADADELTA: An Adaptive Learning Rate Method, *arXiv preprint arXiv:1212.5701* (2012).

[11] Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X. and Chen, X.: Improved Techniques for Training GANs, *Advances in Neural Information Processing Systems 29*, Curran Associates, Inc., pp. 2234–2242 (2016).

[12] Goodfellow, I.: NIPS 2016 Tutorial: Generative Adversarial Networks (2017).