1T-03

小規模データセットでの楽曲生成における LSTM と Transformer の比較

千羽 雄樹 †　　小林 亜樹 †
† 工学院大学情報学部情報通信工学科

1　はじめに

作曲用ソフトやボーカロイドの普及によって，一般人でも作曲できる環境が整いつつある．しかし，作曲するためには音楽理論や音色に関する前提知識が必要なためハードルが高く，思いついたメロディを 1 曲分の長さに仕上げることは困難である．

前提知識なく一片のメロディからの音楽生成にディープニューラルネットワーク（dNN）を使用した自動楽曲生成の研究は盛んであり，系列データ予測器としての LSTM(Long Short Term Memory) による手法や Music Transformer[1] などが提案されて精度を増してきた．その後，複数トラック（複数の楽器）による合奏曲の生成 [2] などへ発展してきている．しかし，人の作曲支援における素材としての主旋律生成を目指す研究は少ない．また，Music Transformer は YouTube から 10000 時間以上の音声を MIDI 形式へ変換した上で学習に用いているが，このような準備には困難があり，多くの研究でデータセット不足が指摘されている．ファインチューニングの活用も考えられるが，利用者の好みを学習データとして反映させようとするとこの点が課題であり，Music Transformer でも少量のデータセットでの性能評価は明らかではない．

そこで本研究では，dNN の LSTM を採用した，主旋律を数音与えるとその続きを生成する「主旋律生成器」を提案する．また，提案する主旋律生成器と Music Transformer で 1 時間程度の小規模のデータセットで学習を行い，生成された楽曲について比較する．

2　提案手法

2.1　概要

dNN の LSTM を採用した，主旋律の初めの数音を MIDI ファイルで与えるとその続きを MIDI ファイルで生成する「主旋律生成器」を実装した．以降の節では，提案手法である主旋律生成器の計算グラフ，曲のデータ形式について説明する．

2.2　主旋律生成器

提案する主旋律生成器は，独自のトークン列として表現された演奏情報を入力に後続の演奏の予測器として振る舞い，音楽生成を行う．トークン系列は主旋律生成に向けて設計し，MIDI ファイル形式の学習データを変換（前処理）して学習させてモデルを得る．システムとしては，入出力とも MIDI 形式で行う．

図 1 は主旋律生成器の計算グラフである．入力層では，トークン列全体を 12 個ずつに区切り，それぞれ数値に変換することで学習可能な値なったものが入力される．出力層では入力された 12 個の入力の次の値が予測として出力される．中間層では 300 ユニットの LSTM 層に加え，その前後にドロップアウト層を追加している．LSTM 層より前のドロップアウト層ではパラメータを 0.25 としており各ノードは 25%の確率で不活性化される．LSTM 層より後のドロップアウト層ではパラメータを 0.5 とした．

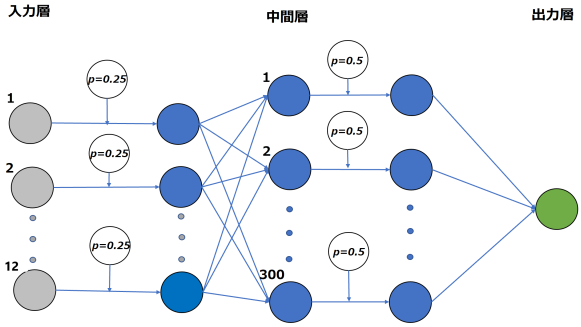

図 1: 主旋律生成器の計算グラフ

2.3　曲データ形式

2.2 節の通り，提案する主旋律生成器は独自のトークン列形式を内部的な入出力に使用する．外部の MIDI 形式から内部トークン形式への変換を前処理と呼ぶ．トークンは，音符を表し，休符も無音の音符として扱う．楽曲の音声波形データを用いる多くの音楽生成研究と異なり，いわば楽譜情報を表現している点が特徴である．例えば，Music Transformer の音楽情報トークンは，「音の始まり」「音の終わり」「時間軸を進める」「次の音の強さ」の 4 つの演奏情報によって表現されている．このため，伴奏付きのメロディを波形的に表現できるが，トークン 1 つあたりの（楽譜としての）演奏情報は少ないため，トークン列は長くなる．一方，提案手法でのトークン列は主旋律のみを扱えるように特化し，その分，トークン 1 つあたりの演奏情報を多く含むようにした．これによって主旋律の後続を生成するのに適していることを狙っている．図 2 に主旋律生成器と Music Transformer のトークン列例を示す．

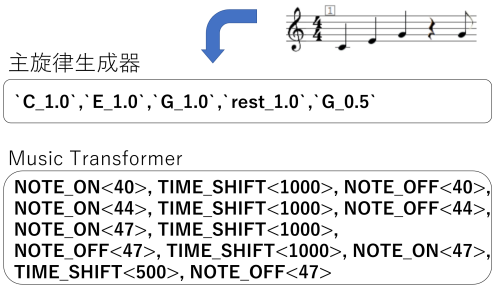

図 2: トークン列（テキスト表現）例

図 2 のような各トークン列は，内部的な数値表現を経て各予測器の計算グラフに入力される．

Comparison of LSTM and Transformer in generating songs on small data sets
†Yuki Chiba †Aki Kobayashi
†Department of Information and Communication Engineering, Kogakuin University

3　楽曲生成

3.1　性能比較

本研究では，提案する LSTM 手法の主旋律生成器と，比較対象の Transformer 手法の Music Transformer で，50 個の MIDI ファイルで学習させ，学習後のモデルに主旋律の初めの数音を MIDI ファイルで与え，その続きを MIDI ファイルで生成させる．生成された 2 種類の楽曲の演奏情報についての比較をし，どちらの楽曲が好きかのアンケートを行うことで，提案手法が主旋律の続きを生成することにおいて有用な手法であるかどうかを確かめる．

3.2　実装

主旋律生成器と Music Transformer は Python 言語で記述されており，提案手法は Keras，Music Transformer は Pytorch による実装である．実行環境は Google Colaboratory を使用した．また，本研究で比較する LSTM 手法については著者が実装した主旋律生成器を使用し，Tranformer 手法は既存実装を使用した [2].

3.3　学習

提案手法と Music Transformer で 50 個の MIDI ファイルの前処理をして演奏情報をトークン列に変換させた結果，主旋律生成器の総トークン列数は 23469，Music Transformer の総トークン列数は 73580 であった．

主旋律生成器は 23469 個のトークンを先頭から 1 つずつずらしながらに 12 トークン毎に区切り，12 トークンを 1 系列データとして扱う 23458 個の系列データを，7:1:2 の割合で訓練データと検証データとテストデータに分割する．Music Transformer は 73580 個のトークンをを先頭から 1 つずつずらしながら 120 トークン毎に区切り，120 トークンを 1 系列データとして扱う 73460 個の系列データを，主旋律生成器と同じ割合で分割する．バッチサイズは 256，エポック数は 1000 としており，このパラメータは主旋律生成器と Music Transformer 共通である．その後 Google Colabolatory 上で学習させた結果，主旋律生成器の学習時間は 17 分 53 秒，Music Transformer の学習時間は 39 分 23 秒となり，提案手法の前処理方法は主旋律生成においてはトークン数が少ない分，高速に学習できると言える．

3.4　結果と考察

学習済みの主旋律生成器と Music Transformer で 3 曲の主旋律を用いてその続きを生成させた．一例として，図 3 にある主旋律を入力したときの生成結果の楽譜を示す．入力した主旋律は図 3 の楽譜の 1 小節目から 4 小節目で青枠で囲まれた部分である．

生成された楽曲について，どちらも調が一定で音程の変化の仕方が不自然では無いように感じた．しかし，主旋律生成器と Music Transformer で生成された楽曲を比較すると，フレーズの最後の音符が Music Transformer より短く，そのあとに続く休符も短かったため，フレーズの終わりの自然さでは提案手法は Music Transformer より劣っていた．

3 曲の主旋律を用いて主旋律生成器と Music Transformer で生成された楽曲それぞれについてどちらが好みかのアンケートを実施した．アンケート参加者は 9 名で，その結果を表 1 に示す．主旋律 2 と 3 では Music Transformer で生成したほうを好んだ回答が多かったが，主旋律 1 では主旋律生成器で生成した楽曲を好んだ回答が多く，入力する主旋律によっては主旋律生成器のほうが人に好まれる楽曲を生成できることが分かった．

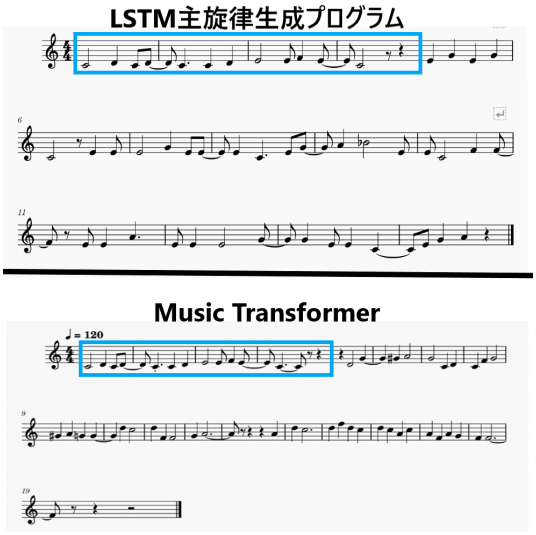

図 3: 主旋律生成器と Music Transformer の前処理方法

表 1: アンケート結果

	主旋律生成器	Music Transformer
主旋律 1	8	1
主旋律 2	0	9
主旋律 3	0	9

楽曲の比較とアンケート結果により，提案手法は複数のフレーズを含んだ主旋律の続きを生成することにおいては有用な手法ではないことが読み取れたが，主旋律の続きを 1 フレーズ分生成することにおいて提案手法は有用な手法であることが確認できた．

4　まとめと今後の課題

本研究では，機械学習モデルの LSTM を利用して，主旋律の続きを自動生成するプログラムを提案した．生成された楽曲について Music Transformer と比較した結果，提案手法はフレーズの終わりの自然さが劣っていたが，主旋律の続きを 1 フレーズ分生成することにおいては提案手法は有用な手法であることが確認できた．

今後の課題として，Music Transformer の前処理を提案手法の前処理に変更することで，より人に好まれる主旋律が生成可能かどうかを確認することが挙げられる．

参考文献

[1] Huang, Cheng-Zhi Anna, et al. ”Music transformer.” arXiv preprint arXiv:1809.04281 (2018).

[2] Dong, Hao-Wen, et al. ”Multitrack Music Transformer: Learning Long-Term Dependencies in Music with Diverse Instruments.” arXiv preprint arXiv:2207.06983 (2022).

[3] gwnndr,MusicTransformerPytorch,https://github.com/gwinndr/MusicTransformer-Pytorch