令和 4 年度
卒 業 論 文

題 目

Music Transformer に基づく 補間フレーズの生成手法

報告者
石山優介

徳島大学理工学部 理工学科
情報光システムコース 情報系
B4 グループ 4 年

目 次

第1章　序論

近年，深層学習に基づく音楽生成技術に注目が集まっている．音楽生成技術の発展により，作曲の知識や楽器の経験が乏しいユーザでも容易に好みの楽曲を作成できるようになり，動画の BGM 生成 [1] や伴奏生成 [2]，作曲支援 [3] など，様々な作曲アプリケーションの基盤技術としてその重要性が高まっている．

Music Transformer[4] は高品質な音楽を生成する深層学習モデルであり，自然言語処理の様々なタスクで高い性能を示すことが報告されている Transformer[5] を音楽生成に適用したものである．Transformer は，従来の畳み込みニューラルネットワーク (Convolutional Neural Network;CNN) や再帰型ニューラルネットワーク (Recurrent Neural Network;RNN) とは異なり，自己注意機構 (Self-Attention) を用いてデータ間の関連度を抽出することで，長期間にわたるデータの特徴表現を学習することができる．また，データの位置情報を Positional Encoding によって付与することにより，学習時の単語ごとの並列処理を可能にしている．MusicTransformer では，Positional Encoding を改良した Performance Encoding[6] を導入し，音楽特有のイベント間の関連性を抽出している．

MusicTransformer は Attention Weight から次のイベントの確率分布を予測し，Softmax 関数を用いて確率的に 1 つのイベントを決定するという処理を繰り返すことにより，入力された音符系列（フレーズ）に続く形で逐次的に音符系列を生成する．しかし，この生成手法は後続フレーズを考慮したフレーズを生成することを考慮しておらず，2 つの既存フレーズを接続するような補間フレーズの生成ができないという問題がある．

本研究の目的は，Music Transformer の Self-Attenion において用いられているマスクに着目し，前側フレーズの続きを予測する際に後側フレーズを参照できるようにすることで，補間フレーズを生成できないという問題を解決する．作曲の過程において，作曲済みの断片的なフレーズ同士を自然に接続できる補間フレーズを作曲・編曲する機会は頻繁にあるため，補間フレーズを自動生成する提案手法のニーズは高い．生成した補間フレーズについて，真の補間フレーズとの類似性を評価する実験を行い，提案手法の有効性を確認した．

本稿の構成は下記の通りである．まず，第 2 章では，関連研究とその問題点について述べる．次に第 3 章では，本研究のベースラインとなった Music Transformer とその問題点について述べる．続いて第 4 章では，本研究の提案手法を述べ，第 5 章では行った従来手法との比較実験の結果と考察を述べる．最後に第 6 章では，本稿のまとめと今後の課題について述べる．

第2章　関連研究

本章では，本研究の関連研究として，音符系列の潜在空間を学習することで異なるフレーズ間を補間する MusicVAE [7] および，Music Transformer[4] を 2 度使用することで後側のフレーズに適した中間フレーズを生成する清水の手法 [8] について述べる．

2.1　MusicVAE

MusicVAE の要素技術であるオートエンコーダ [9]，変分オートエンコーダ [10] について説明する．

2.1.1　オートエンコーダ

オートエンコーダとは，入力情報を低次元の隠れ層の空間に写像して特徴抽出を行うエンコーダ f_{encode} と，抽出された特徴を再度元の次元に戻すことで入力情報の復元を目指すデコーダ f_{decode} から構成されるエンコーダ・デコーダモデルである．オートエンコーダは，入力データそのものを教師データとして用いて学習を行う．これは教師なし学習であり，式 (2.1) の平均二乗誤差 (Mean Squared Error;MSE) を最小化することで行われる．ここで X はオートエンコーダの入力である．

$$L = MSE\left(f_{decode}\left(f_{encode}\left(X\right)\right), X\right) \tag{2.1}$$

2.1.2　変分オートエンコーダ

変分オートエンコーダ (Variational Auto-Encoder;VAE) は，オートエンコーダと異なり，その潜在変数分布 $p\left(z\right)$ は標準ガウス分布 $z \sim N\left(\mu, \sigma^2\right)$ に従わなければならない．したがって，VAE におけるデータ生成の観点からは，以下の 2 段階のプロセスでデータポイントが生成される．

1. 事前分布 $p\left(z\right) : z \sim p\left(z\right)$ から潜在コードをサンプリングする．

2. 潜在コード $X \sim p(X \mid z)$ をデコードしてデータポイントを生成する．

この性質を利用するため，VAE の損失関数は，周辺尤度 $p(X)$ の証拠下限を最大化するもので，式 (2.2) が成り立つ．

$$L = E_{q(z|X)}\left[\log p(X \mid z)\right] - D_{KL}\left(q(z \mid x) \,||\, p(z)\right) \tag{2.2}$$

式 (2.2) の第 1 項 $E_{q(z|X)}\left[\log p(X \mid z)\right]$ は対数尤度 $\log p(X \mid z)$ の事後分布 $q(z \mid X)$ に対する期待値であり，事後分布にしたがって得られた潜在変数と入力データがどれだけ類似しているかを表す数値となるため，式 (2.1) に類似していることがわかる．VAE の損失において追加された第 2 項 $D_{KL}\left(q(z \mid x) \,||\, p(z)\right)$ は KL ダイバージェンスであり，これは 2 つの確率分布の差を測定するために一般的に使用される．ここでは，事後分布 $q(z \mid X)$ が事前分布 $p(z)$ に近づくことを確認しており，事前分布は一般的に標準ガウス分布 $N\left(\mu, \sigma^2\right)$ である．つまり，オートエンコーダはエンコードされた潜在変数の分布に制約を与えないが，VAE は潜在変数分布 $p(z)$ が標準ガウス分布 $z \sim N\left(\mu, \sigma^2\right)$ に従わなければならないため，正則化されたオートエンコーダと考えることができる．したがって，エンコード-デコードのプロセスも以下のように若干異なる．

- エンコードの際，1 つの潜在ベクトルだけをエンコードするのではなく，基礎となる分布の平均 μ と標準偏差 σ を表す 2 つの潜在ベクトルをエンコードする．
- デコードの際，まず分布から潜在ベクトルをサンプリングする (正確には，再パラメータ化トリック $\epsilon \sim N(0, I), z = \mu + \sigma \cdot \epsilon$ によって行われる)．そして，潜在ベクトルをデコーダに入力し，入力を再構成する．

オートエンコーダの潜在空間構造には，制約条件がないため，データ範囲が散らばり，まとまりがない．それに対して VAE では，潜在変数分布 $p(z)$ が標準ガウス分布 $z \sim N\left(\mu, \sigma^2\right)$ に従わなければならないため，潜在空間においてデータごとにまとまりがある．これにより狙ったサンプリングが行いやすく，オートエンコーダより補間フレーズの生成に向いていることが分かる．

2.1.3　MusicVAE のエンコーダとデコーダ

MusicVAE のエンコーダとデコーダには長･短期記憶 (Long short-term memory;LSTM)[11] が用いられている．エンコーダ $q(z \mid X)$ は入力系列を処理し，隠れ状態の系列を生成す

る．平均 μ と標準偏差 σ は最終的な隠れ状態 h_T を用いて 2 つの別々のフィードフォワードネットワーク $\mu = f_1(h_T), \sigma = f_2(h_T)$ によって導出される．デコーダ $p(X \mid z)$ は，まず，サンプリングされた潜在ベクトル z を用いて，デコーダ LSTM の初期状態を設定する．そして，初期状態から出力系列を自己回帰的に生成する．学習時には，出力系列は入力系列を再構成するように学習される．メロディ列は (T,130) 行列で表現され，130 次元の出力空間は 128 の MIDI ピッチに対応する 128 個のノートオントークンと，単一トークンであるノートオフ，レスト（何も弾かない）から構成されている．

MusicVAE に Vanilla RNN(通常の LSTM) を使用した場合，系列全体を単一の潜在ベクトルに圧縮するため，非常に厳しいボトルネックをもたらす．これにより，短い系列のメロディーであれば適切な精度で復元できるが，長い系列では精度が低下することが確認されている．そこで，この研究では階層型リカレント変分オートエンコーダを導入している．階層型リカレント変分オートエンコーダの概要を図 2.1 に示す．

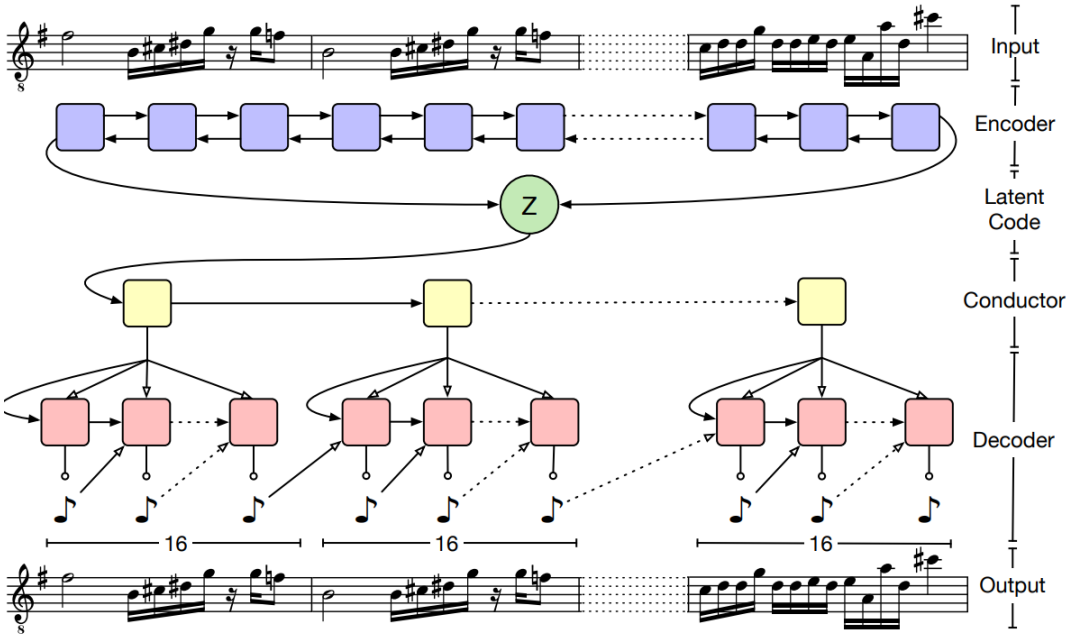

図 2.1: 階層型リカレント変分オートエンコーダの概要図．[7] より引用

これは，各小節ごとに中間的な潜在ベクトルを生成し，生成された中間的な潜在ベクトルに基づいて各小節の出力を行うモデルである．長い系列を圧縮すると潜在状態の影響の消失が起こるため，短い系列での中間的な潜在ベクトルの生成を行うことで，消失問題を緩和することができる．さらに，潜在ベクトルから得られる特定の小節の中間潜在ベクト

ルだけに基づいて出力を生成することにより，潜在状態を無視しないように生成することができ，潜在分布の効率的な学習につながる．

2.1.4　生成

2 つのメロディ A と B の間の補間は以下の手順で行う．

1. A と B の潜在的なコード z_A, z_B を取得する．
2. z_A と z_B の点を潜在空間上で「スライド」させ，補間経路上の N 個の潜在コード，すなわち $z_\alpha = z_A + \alpha\,(z_B - z_A)$ を取得する．

その結果，潜在空間上で補間すると，データ空間上で補間した場合に比べて，音源メロディから対象メロディへの変換がよりスムーズで一貫したものになることが実証された．このことは，学習した潜在分布が音楽断片の構造に関連した意味のあるコンパクトな情報を捉えることが可能であることを証明している．

2.2　清水の手法

清水の手法では Music Transformer を 2 度用いることで，2 つのフレーズ間をつなぐ中間フレーズを生成する．

2.2.1　システム概要

以下，元となる曲のフレーズの前側部分を $A = \{a_1, \ldots, a_{L_A}\}$，フレーズの後側部分を $C = \{c_1, \ldots, c_{L_C}\}$，生成された中間部分を $\hat{B} = \{\hat{b}_1, \ldots, \hat{b}_{L_B}\}$ とする．

1. 前側のフレーズ A を Music Transformer に入力し，新たな中間となるフレーズ $\hat{B}_1$ を生成する．
2. 生成された中間となるフレーズ $\hat{B}_1$ をもう一度 Music Transformer に入力し，さらに新たなフレーズ $\hat{C}_1$ を生成する．
3. $\hat{C}_1$ と本来のフレーズの後側である C を比較し，類似度を測る．
4. 1 から 3 の処理を n 回繰り返し，最も類似度の高い $\hat{C}_i$ を生成した $\hat{B}_i (i = 1, \ldots, n)$ を抽出する．

5. 抽出された $\hat{B}_i$ を，本来のフレーズの前側である A, 後側である C と結合し，出力する．

以上の手順により，前後を考慮したフレーズの生成を行う．清水の手法の概要を図 2.2 に示す．

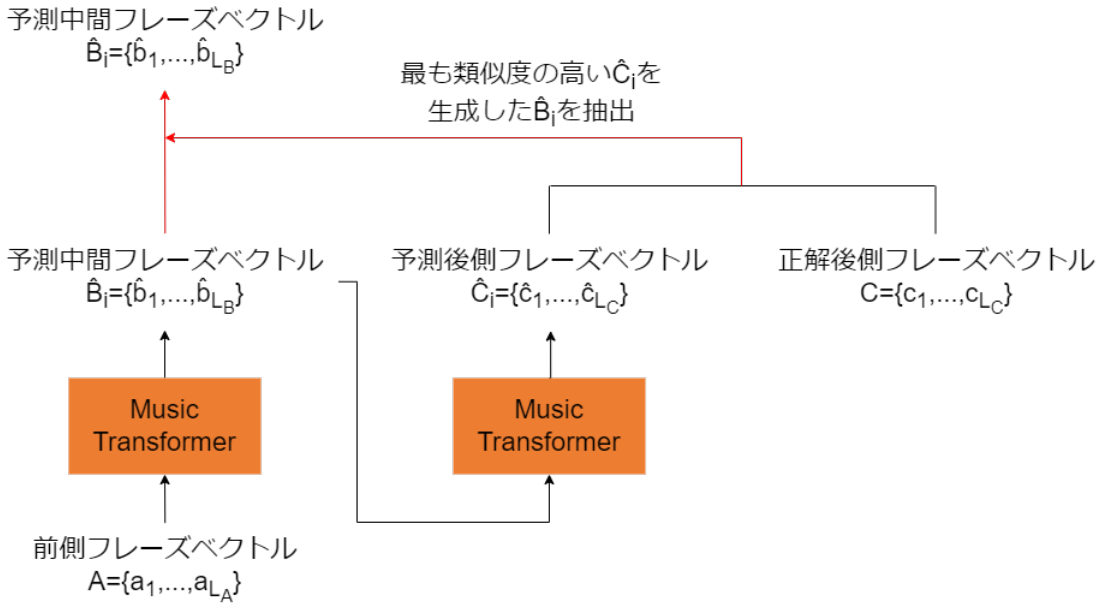

図 2.2: 清水の手法の概要図

清水の手法で用いられる Music Transformer において，生成される音符系列全体の長さは $L = 1024$，入力する 音符系列 A の長さは $L_A = 256$ で学習される．音符系列 A を Music Transformer に入力すると，音符系列 A とそれに続く形で，中間の音符系列 $\hat{B}_i(i = 1, \ldots, n)$ を生成し，2 つを結合したベクトルが出力される．よって，音符系列 $\hat{B}_i$ の長さは，$L_{B_i} = L - L_A = 768$ となる．Music Transformer 一連の生成を $MusicTransformer$ とし，式 (2.3) に示す．

$$MusicTransformer(A) = A + \hat{B}_i \tag{2.3}$$

その後，生成された $\hat{B}_i$ をもう一度 Music Transformer に入力する．その際，$\hat{B}_i$ の先頭から長さ $L_{B_i} = 256$ の音符系列を抽出し，Music Transformer に入力する．これを式 (2.4) に示す．

$$MusicTransformer(B_i) = \hat{B}_i + \hat{C}_i \tag{2.4}$$

1 度目の生成同様，生成された音符系列 $\hat{C}_i$ の長さは $L_{C_i} = L - L_{B_i} = 768$ となる．そして，$\hat{B}_i$ のように $\hat{C}_i$ の先頭から長さ $L_{C_i} = 256$ の音符系列を抽出する．

音符系列 $\hat{B}_i, \hat{C}_i$ をそれぞれ 1 つずつ生成した後，生成した $\hat{C}_i$ と正解である音符系列 C をコサイン類似度を用いて比較する．したがって，コサイン類似度は式 (2.5) のようになる．この時，C の音符系列の長さは $L_C = 256$ とする．

$$\cos(\hat{C}_i, C) = \frac{\hat{C}_i \cdot C}{||\hat{C}_i|| ||C||} \tag{2.5}$$

$$= \frac{\sum_{n=1}^{L_C} \hat{C}_{in} C_n}{\sqrt{\sum_{n=1}^{L_{\hat{C}_i}} \hat{C}_{in}^2} \sqrt{\sum_{n=1}^{L_C} C_n^2}} \tag{2.6}$$

$i = 1$ であればそれを最良値とし，それ以降は値が $d_{i+1} \geq d_i (-1 \leq d_i, d_{i+1} \leq 1)$ となった時に更新され，その最良値が保存される．

以上の処理を n 回繰り返す．最も d が高かった $\hat{C}_i$ を生成した $\hat{B}_i$ を抽出し，それを本来のフレーズ A, C の中間に加えることで，長さ $L_{A+B+C} = L_A + L_B + L_C = 768$ の新たなフレーズを得る．

2.3　問題点

MusicVAE は単一の音を生成することによる補間フレーズの作成を行うことはできるが，2 つ以上の音を同時に鳴らす和音を生成できないという問題がある．また，清水の手法は前側フレーズから生成された中間フレーズから予測された後側フレーズと本来の後側フレーズを比較することで，生成された中間フレーズの中から適したフレーズを選択しているが，正解の後側フレーズは比較に用いられているだけで，実際に中間フレーズを生成する際には考慮されていないという問題がある．

第3章　Music Transformer

本章では，MusicVAE[7] では不可能だった和音によって構成されるフレーズを生成できるモデルであり，清水の手法 [8] に用いられていた Music Transformer[4] について述べる．

Music Transformer は，自然言語処理タスクで高い精度で学習できることが報告されている Transformer[5] を音楽に適用した手法で，音符系列内の要素間の関連度を捉える Self-Attention 構造を持つエンコーダのみのモデルである．エンコーダは，入力されたフレーズの音符系列から中間表現を獲得し，それを線形変換することで，入力されたフレーズに続く新しいフレーズの音符系列を予測して出力する．

Music Transformer は，入力された音符系列の最初から逐次的に次のイベントを予測して新たなフレーズを生成するモデルである．具体的には，音符系列の最初のイベントを予測し，続いて，予測したイベントを再びエンコーダに入力して 2 つ目のイベントを予測する．続いて，予測した最初から 2 つ目までのイベントを結合した音符系列を再びエンコーダに入力して 3 つ目のイベントを予測する，といった処理を繰り返すことで音符系列の予測を行う．Music Transformer の概要を図 3.1 に示す．

3.1　入力部

入力部では，MIDI データの前処理と埋め込み処理が行われる．前処理には Oore らによって提案された Performance Encoding[6]，埋め込み処理には埋め込み層と Positional Encoding が用いられる．

3.1.1　Performance Encoding

Music Transformer では，MIDI データを Transformer モデルで処理可能なベクトルに変換するために Performance Encoding を適用する．これは，MIDI データを音の鳴り始めを表す 128 個の NOTE_ON イベント，音の鳴り終わりを表す 128 個の NOTE_OFF イベント，10ms 刻みの時間進行を表す 100 個の TIME_SHIFT イベント (10〜1000ms)，音

の強弱を表す 32 個の VEROCITY イベント の合計 388 個の語彙で構成されたイベントベクトルに変換する処理である．図 3.2 に Performance Encoding の例を示す．図 3.2 の左図はピアノ演奏を視覚化したものである．右図は上から下に実行されるイベントである．C メジャーコードが C,E,G の順で 0.5 秒おきに演奏され 2 秒まで続き，そこで音が一度なくなる．2.5 秒から F の音が 0.5 秒間演奏され，3 秒時点で全て終了する．C メジャーコードは 80 の速度，F は 100 の速度で演奏される．

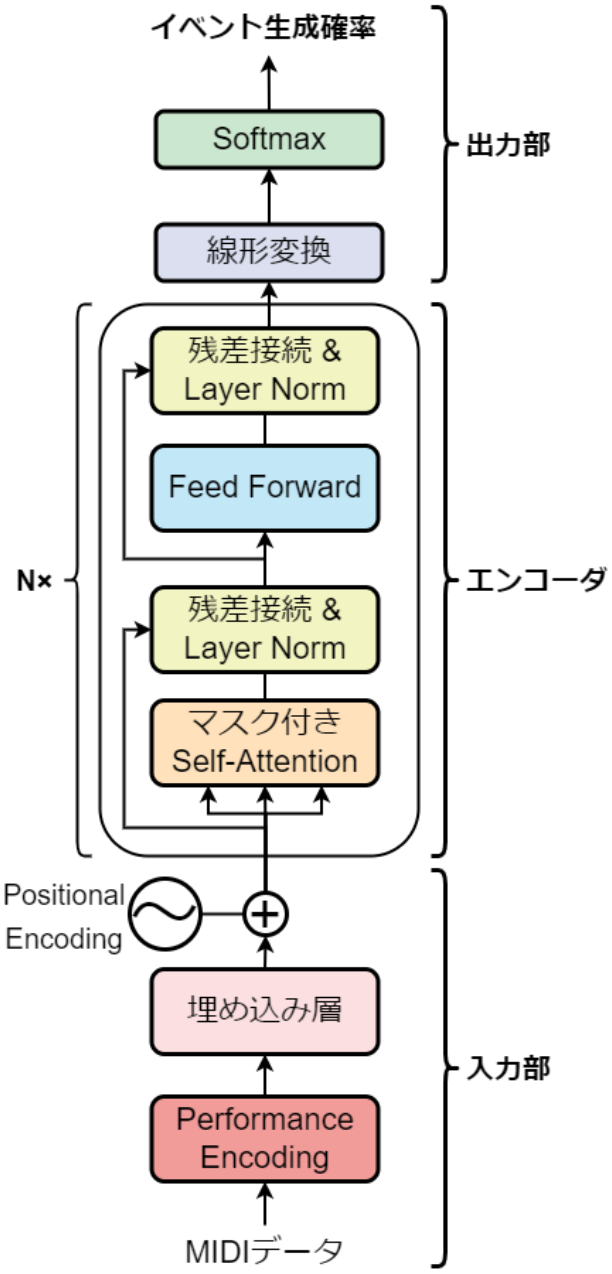

図 3.1: Music Transformer の概要図 [4]

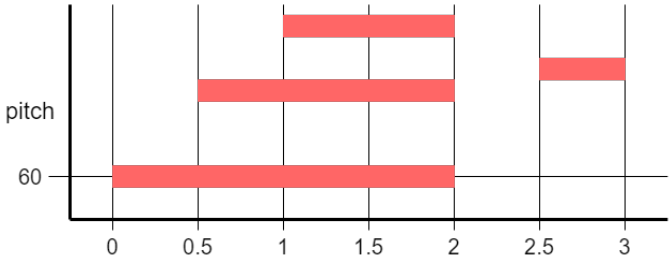

図 3.2: Performance Encoding の例

3.1.2　埋め込み層と Positional Encoding

Performance Encoding で生成されたイベント系列をエンコーダに入力する際に，埋め込み層でイベント系列を埋め込み表現行列に変換し，Positional Encoding によってイベントの系列情報を付与する．具体的には，入力イベント系列の埋め込み表現行列に対して系列における各イベントの絶対的な位置情報をエンコードした行列 PE を加える．PE の各成分は異なる周波数の sin，cos 関数を用いて式 (3.1), 式 (3.2) により算出したものである．

$$PE\left(pos, 2i\right) = \sin\left(pos/10000^{2i/d_{model}}\right) \tag{3.1}$$

$$PE\left(pos, 2i+1\right) = \cos\left(pos/10000^{2i/d_{model}}\right) \tag{3.2}$$

ここで，d_{model} は入力イベントの埋め込み次元，pos はイベントの位置，i は各成分の次元を表す．イベント埋め込み表現行列に PE を加えたものが，第 1 層目のエンコーダレイヤの入力となる．

3.2　エンコーダ

エンコーダレイヤは，下位のサブレイヤから順に，予測するイベントより前方に位置するイベント間のみの関連度捉えるマスク付き Self-Attention，位置ごとのフィードフォワードネットワーク (Feed Forward Network; FFN) の 2 つのサブレイヤで構成されている．

各サブレイヤ間では，残差接続 [12] を行った後に Layer Normalization [13] が適用される．Layer Normalization を適用する関数を $LayerNorm$，下位のサブレイヤからの出力を x，現在のサブレイヤの処理を行う関数を $SubLayer$ とすると，$LayerNorm\left(x + SubLayer\left(x\right)\right)$ が現在のサブレイヤの出力となる．

3.2.1　Transformer における Self-Attention

以下，系列の長さ，隠れ状態のサイズ，ヘッドの数をそれぞれ L, D, H で表す．Transformer における Self-Attention では，エンコーダの内部状態系列 $X = (x_1, \ldots, x_L)$ を Query：$Q = XW^Q$，Key：$K = XW^K$，Value：$V = XW^V$ に変換する．W^Q, W^K, W^V はそれぞれ $D \times D$ の正方行列である．次に，各 $L \times D$ の Query, Key, Value が H によって，$L \times D_h$ または，Attention Head に分割された後，h でインデックス付けされることで，$D_h = \frac{D}{H}$ の次元でモデルが様々な部分に注目可能になる．スケーリングされた 内積 Attention は，各ヘッドに対してベクトル出力列を式 (3.3) により算出される．

$$Z^h = Attention\left(Q^h, K^h, V^h\right) = softmax\left(\frac{Q^h K^{h\mathsf{T}}}{\sqrt{D_h}}\right) V^h \tag{3.3}$$

3.2.2　Music Transformer における Self-Attention

Music Transformer における Self-Attention では，Shaw らが提案した相対位置表現 [14] を導入し，系列内の 2 つのイベントの位置がどれだけ離れているかによって Attention を考慮できるようになる．これには，Query と Key の位置がそれぞれ i_q と j_k にある場合の 2 つの距離 $r = j_k - i_q$ に対応する埋め込みを持つ形状 (H, L, D_h) の相対位置埋め込み E^r の学習が含まれる．埋め込みは，距離 $-L+1$ から 0 の順に並べられ，各ヘッドに対して個別に学習される．Shaw らの研究では，相対的な埋め込みが Query と相互作用し，S^{rel} という $L \times L$ 次元のロジットの行列を生成し，各ヘッドの Attention の確率は式 (3.4) となる．

$$RelativeAttention = softmax\left(\frac{QK^{\mathsf{T}} + S^{rel}}{\sqrt{D_h}}\right) V \tag{3.4}$$

この手法により，S^{rel} を計算するためのメモリ使用量を大幅に改善しながら，相対距離情報を考慮して Attention の計算を行える．Shaw らの研究では，各ヘッドごとに，全ての Query，Key 間の相対距離に対応する埋め込みを含む，形状 (L, L, D_h) の中間テンソル R をインスタンス化する．そして，Q は $(L, 1, D_h)$ のテンソルに再形成され，$S^{rel} = QR^{\mathsf{T}}$ となる．これにより，$O\left(L^2 D\right)$ の全体的な空間の複雑さが発生し，長い系列の適用のみに制限される．

3.2.3　Relative Attention の効率的な実装

Shaw らの手法では，Relative Attention の実装を改善し，中間メモリ要件を $O\left(L^2D\right)$ から $O\left(LD\right)$ に減らした．Q と相対位置埋め込みである E^r を直接乗算すれば，QR^T から必要な項は全て利用可能であることが分かる．$QE^{r\mathsf{T}}$ を計算した後，その (i_q, r) エントリは，位置 i_q の Query と相対距離 r が埋め込みの内積が含まれる．ただし，式 (3.4) の行列 S^{rel} の各相対ロジット (i_q, j_k) は，QK^T のインデックスと一致するように，位置 i_q の Query と相対距離 $j_k - i_q$ の埋め込みとの内積にする必要がある．したがって，$QE^{r\mathsf{T}}$ を skew することで，相対ロジットを正しい位置に移動させる必要がある．skew については次項で詳しく説明する．この手法による実装と，Shaw らの手法による実装では時間計算量はどちらとも $O\left(L^2D\right)$ だが，実際には長さ 650 で 6 倍高速になる．Shaw らの手法と Music Transformer での手法の時間計算量，$D_h = 64$ と仮定した際に 16GB メモリの GPU に収まる最大系列長，および系列長 L のときの 1 層 1 ヘッドのメモリ使用量 (単位：MB) の比較を表 3.1 に示す．

表 3.1: Relative Attention の比較．[4] より引用し一部改変

実装	時間計算量	最大系列長	$L = 650$	$L = 2048$	$L = 3500$
Shaw et al. (2018)	$O\left(L^2D + L^2\right)$	650	$108 + 1.7$	$1100 + 16$	$3100 + 49$
Music Transformer	$O\left(LD + L^2\right)$	3500	$0.17 + 1.7$	$0.52 + 16$	$0.90 + 49$

skew アルゴリズム

skew とは，絶対値 $\times$ 相対値である (i_q, r) のインデックス付き行列を，絶対値 $\times$ 絶対値である (i_q, j_k) のインデックス付き行列に変換する処理である．行のインデックス i_q は同じままで，列のインデックス j_k が $j_k = r - (L-1) + i_q$ の式に従いシフトされる．図 3.3 の下部は skew アルゴリズムの説明であり，このアルゴリズムは上段の時間計算量 $O\left(L^2D\right)$ のアルゴリズムと異なり，R のインスタンス化を必要としない．灰色はマスクまたはパディングされた位置，各色は相対距離の異なる位置を示す．

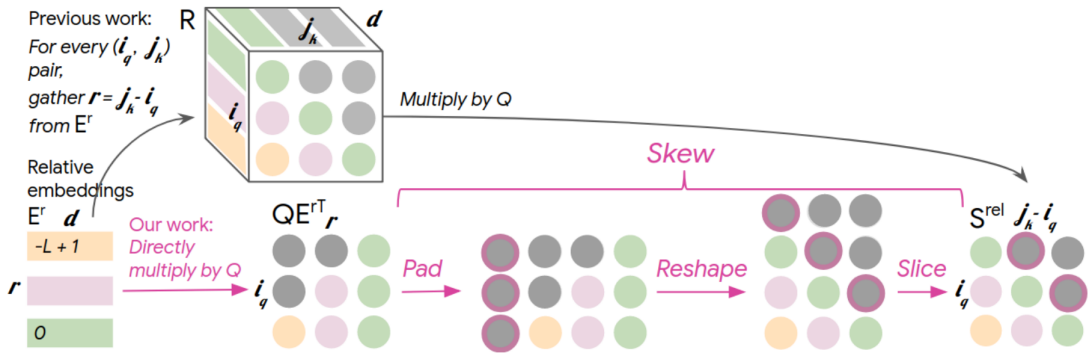

図 3.3: Relative Global Attention．[4] より引用

図 3.3 が示す手順を以下に示す．

1. 左端の列の前に長さ L のダミー列ベクトルを加える．
2. 行列の形状を $(L+1, L)$ に変更する．(このステップでは，Numpy 形式での行優先の順序を想定)
3. 行列をスライスして，最後の l 行と全ての列を保持することで作成される絶対値 $\times$ 絶対値である (L, L) のインデックス付き行列が S^{rel} である．

また，非常に長い系列では，ベースライン Transformer でさえ 2 次関数的にメモリを必要とするため，実用的ではない．例えば，Wikipedia や画像生成 [15] において，入力系列を重複しないブロックに分割することで Local Attention が利用されてきた．そして，図 3.4 の右上に示されているように，各ブロックは自身とその前のブロックに注目する．

局所的な場合においても Relative Attention を使用するためには，右ブロックは図 3.3 の大域的な場合と同じ構成だが，L^2 ではなく非常に小さい $\left(\frac{L}{M}\right)^2$ であることに注意する必要がある．この時，M はブロック数，N は結果のブロック長とする．左ブロックは，-1 (右上) から $-2N+1$ (左下) まで相対インデックスでマスクされない．したがって，局所的な場合の学習済み E^r は $(2N-1, N)$ の形状を持つ．

大域的な場合と同様に，最初に QE^r を計算し，図 3.4 に示すように QK^T と同じインデックスを持つように skew するために以下の手順を実行する．

1. 右端の列の後に長さ N のダミー列ベクトルを加える．
2. 行列を平坦化し，長さ $N-1$ のダミー行を加える．

3. $(N+1, 2N-1)$ の形状になるよう行列を再形成する．

4. 行列をスライスして最初の N 行と最後の N 列のみを保持し，(N, N) 行列を作成する．

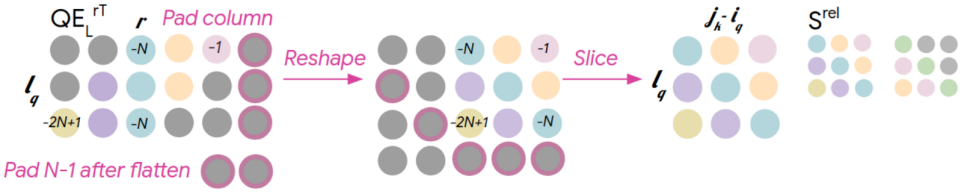

図 3.4: Relative local attention:左図は skew の手順，右図は S^{rel} の望ましい構成を示している．[4] より引用

3.2.4　マスク付き Self-Attention

マスク付き Self-Attention では，エンコーダの内部状態系列を Query，Key，Value に用いる．しかし，推論時には予測するイベントより後で生成されるイベントを知ることはできないため，前方のエンコーダ系列の内部状態のみを用いる．また，学習時も推論時との整合性を保つため，予測するイベントより後方のイベント間の関連度を考慮しないマスク付き Self-Attention を用いる．具体的には下記のように変更を加える．各ヘッドの Attention 出力 $Z = (z_1, \ldots, z_L)$ の要素 z_i は式 (3.5) で定義できる．

$$z_i = \sum_{j=1}^{n} \alpha_{ij} x_j W^V \tag{3.5}$$

重み係数 α_{ij} は softmax 関数を用いて以下の通り計算される．

$$\alpha_{ij} = \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{n} \exp\left(e_{ik}\right)} \tag{3.6}$$

また，e_{ij} は以下の通り計算される．

$$e_{ij} = \frac{\left(x_i W^Q\right)\left(x_j W^K\right)^{\mathsf{T}} + S^{rel}}{\sqrt{D_h}} \tag{3.7}$$

式 (3.7) を式 (3.8) に変更する．

$$e_{ij} = \begin{cases} \dfrac{\left(x_i W^Q\right)\left(x_j W^K\right)^{\mathsf{T}} + S^{rel}}{\sqrt{D_h}} & (i \geq j) \\ -\infty & (otherwise) \end{cases} \tag{3.8}$$

式 (3.7) を式 (3.8) に変更することで，後方に位置するイベントとの関連度を表す重み係数 $\alpha_{ij}\,(i<j)$ は式 (3.9) のように 0 となる．

$$\alpha_{ij} = \begin{cases} \dfrac{\exp(e_{ij})}{\sum_{k=1}^{n}\exp(e_{ik})} & (i \geq j) \\ 0 & (otherwise) \end{cases} \tag{3.9}$$

3.2.5　Feed Forward サブレイヤ

各ヘッドの Attention 出力は連結され，線形変換されて，$L \times D$ 次元の行列である Z となる．次に，残差接続を行った後に Layer Normalization が適用される．その後，Feed Forward (FF) サブレイヤにより，前の Attention サブレイヤからの出力 Z を入力とし，深さ D の次元で以下の計算を行う．W_1, W_2, b_1, b_2 はその 2 層の重みとバイアスである．

$$FF(Z) = ReLU(ZW_1 + b_1)W_2 + b_2 \tag{3.10}$$

最後に，再び残差接続を行った後に Layer Normalization が適用され，出力部へと入力される．

3.3　出力部

Music Transformer では，最終エンコーダレイヤの出力 $(h_1, \ldots, h_n)$ を重み行列 $W^{out} \in \mathbb{R}^{d_{model} \times d_{out}}$ により線形変換し，その後 softmax 関数を用いて各イベントの生成確率を得る．ここで，d_{out} はイベントの語彙サイズである．すなわち，出力イベント y_i の生成確率分布は以下の通りになる．

$$p(y_i) = softmax\left(h_i W^{out}\right) \tag{3.11}$$

$$softmax(o) = \frac{\exp(o_k)}{\sum_{k=1}^{d_{out}}\exp(o_k)} \tag{3.12}$$

出力イベント系列 $y_1, \ldots, y_m$ は，各イベントの生成確率分布に基づき，greedy アルゴリズム [16] やビーム探索 [17] 等により生成する．

3.4　問題点

Music Transformer は，入力された音符系列からそれに続く新たな音符系列を逐次的に出力するモデルであり，予測するイベントより後方に位置するイベントとの関連度を捉え

ることができないため，フレーズ間をつなぐ補間フレーズの生成に適していないという問題がある．

第4章　提案手法

本章では，本研究の目的である前後フレーズを考慮した補間フレーズの生成を実現する提案手法について述べる．

4.1　従来手法からの変更点

改善方法として，マスク付き Self-Attention に変更を加えることを提案する．具体的には，前側フレーズの系列長を L_A，中間フレーズの系列長を L_B，後側フレーズの系列長を L_C としたとき，式 (3.8) を式 (4.1) に変更する．

$$e_{ij} = \begin{cases} \dfrac{\left(x_i W^Q\right)\left(x_j W^K\right)^{\mathsf{T}} + S^{rel}}{\sqrt{D_h}} & (i \geq j \parallel j \geq L_A + L_B) \\ -\infty & (otherwise) \end{cases} \tag{4.1}$$

式 (3.8) を式 (4.1) に変更することで，後方に位置する，かつ後側フレーズでないイベントとの関連度を表す重み係数 $\alpha_{ij}\,(i < j \ \&\ j < L_A + L_B)$ は式 (4.2) のように 0 となる．

$$\alpha_{ij} = \begin{cases} \dfrac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{n} \exp\left(e_{ik}\right)} & (i \geq j \parallel j \geq L_A + L_B) \\ 0 & (otherwise) \end{cases} \tag{4.2}$$

$L_A = 3$，$L_B = 3$，$L_C = 3$ とした場合のマスクの例を図 4.1 に示す．左図が従来手法，右図が提案手法のマスクとなっており，灰色はマスクされた要素の位置を示す．l 行目は $(l+1)$ 番目のイベントを推測する際のマスクとなっており，従来手法では l 番目以前のイベントとの関連度しか捉えられない状態だが，提案手法では l 番目以前のイベントと後側 L_C つのイベントとの関連度を捉えられる状態であることを表している．

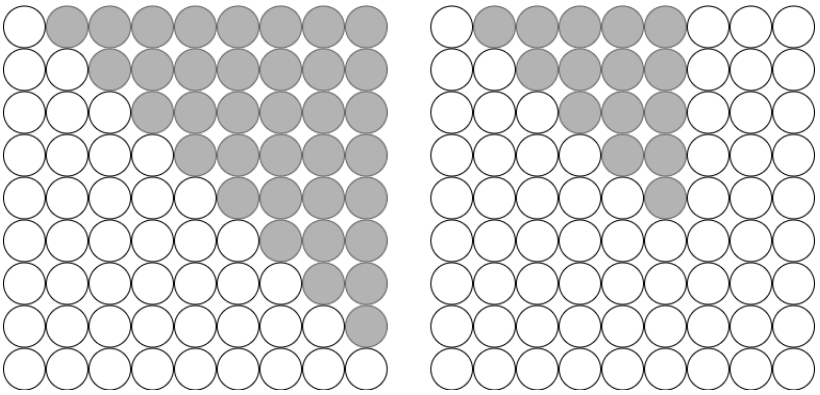

図 4.1: 従来手法と提案手法のマスク形状

これにより，予測するイベントより前方に位置するイベントと後側フレーズのイベントの両方の関連度を捉えられるようになり，前後フレーズを考慮した補間フレーズの生成に適したモデルとなる．

4.2　生成

生成フェーズでは，前側フレーズと後側フレーズを入力することで，前側フレーズと補間フレーズと後側フレーズを結合した音符系列を出力する．前側フレーズの系列長を L_A，後側フレーズの系列長を L_C，生成される補間フレーズの系列長を L_B，マスクされるダミーの系列長を L_X とし，前側フレーズを $A = \{a_1, \ldots, a_{L_A}\}$，後側フレーズを $C = \{c_1, \ldots, c_{L_C}\}$，生成される補間フレーズを $B = \{b_1, \ldots, b_{L_B}\}$，マスクされるダミー系列を $X = \{x_1, \ldots, x_{L_X}\}$ とすると，以下の手順で生成が行われる．

1. 前後フレーズ A, C をダミー系列 X で結合した系列 A, X, C をモデルに入力する．
2. ダミー系列 X にマスクを付け，x_1 の位置にあたる b_1 を生成する．
3. 生成された b_1 を x_1 に代入し，新たな系列 $A, b_1, X_{2\sim L_x}, C$ をモデルに入力する．
4. ダミー系列 $X_{2\sim L_x}$ にマスクを付け，x_2 の位置にあたる b_2 を生成する．
5. 生成された b_2 を x_2 に代入し，新たな系列 $A, b_1, b_2, X_{3\sim L_x}, C$ をモデルに入力する．
6. 2〜5 の処理を L_x 回繰り返し，A, B, C を出力する．

以上の手順を $L_A = 3, L_C = 3, L_B = 3, L_X = 3$ とした場合の生成の例を図 4.2 に示す.

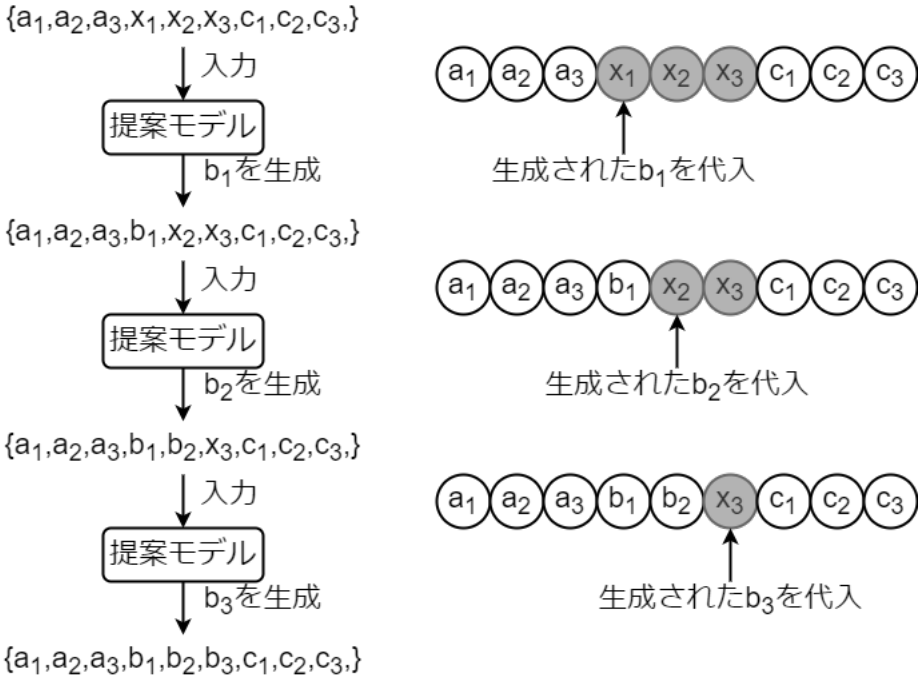

図 4.2: 提案手法における補間フレーズ生成の流れと生成時のマスク位置

図 4.2 は左図が提案手法の補間フレーズ生成の流れ，右図が生成時のマスクの位置を灰色で示している．はじめに，入力した前後フレーズをダミー系列で結合した系列 $\{a_1, a_2, a_3, x_1, x_2, x_3, c_1, c_2, c_3\}$ がモデルに入力される．このとき，モデルでは x_1, x_2, x_3 にマスクが付き，前後のフレーズのみを考慮して x_1 の位置にあたる b_1 が生成される．そして，生成された b_1 が x_1 に代入され，新たな系列 $\{a_1, a_2, a_3, b_1, x_2, x_3, c_1, c_2, c_3\}$ がモデルに入力される．このとき，モデルでは x_2, x_3 にマスクが付き，前後のフレーズと b_1 を考慮して x_2 の位置にあたる b_2 が生成される．そして，生成された b_2 が x_2 に代入され，新たな系列 $\{a_1, a_2, a_3, b_1, b_2, x_3, c_1, c_2, c_3\}$ がモデルに入力される．このとき，モデルでは x_3 にマスクが付き，前後のフレーズと b_1, b_2 を考慮して x_3 の位置にあたる b_3 が生成される．最後に，生成された b_3 が x_3 に代入され，前後のフレーズが補間フレーズによって接続された音符系列が出力される．

第5章　実験

本実験では，従来手法として Self-Attention のマスクを変更しない Music Transformer を用い，提案手法と従来手法による補間フレーズの生成精度を評価する．

5.1　データセット

本実験では，Piano-e-Competition dataset[18] から 2,287 曲を学習用データとして，254 曲をテスト用データとして用いる．

5.2　生成

従来手法では，系列長 512 の前側フレーズを入力し，それに続く系列長 1,024 の補間フレーズを生成する．最後に前側フレーズ，生成された補間フレーズと系列長 512 の後側フレーズを結合することで音符系列を生成する．提案手法では，系列長 512 の前側フレーズ，系列長 1,024 のダミーフレーズ，系列長 512 の後側フレーズを結合した系列を入力する．入力系列のダミーフレーズ部分と置き換わる系列長 1,024 の補間フレーズを生成し，音符系列を生成する．

5.3　定量的評価

定量的評価では 254 曲のテスト用データに対して提案手法及び従来手法によって生成された補間フレーズと本来の後側フレーズのクロマベクトルのコサイン類似度を比較することにより，後側フレーズを考慮した生成ができているかを評価する．クロマベクトルとは，12 音階及びオクターブ違いも含めた音階ごとの周波数の振幅強度を特徴量としたベクトルである．式 (5.1) にクロマベクトル a, b 間のコサイン類似度 $\cos(a, b)$ の定義を示す．コ

サイン類似度とは，2 つのベクトルの類似性を表す尺度である．

$$\begin{aligned}\cos(a,b) &= \frac{a\cdot b}{\|a\|\|b\|} \\ &= \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} b_i^2}}\end{aligned} \tag{5.1}$$

5.4　実験結果

254 曲のテスト用データにおいて生成された補間フレーズと後側フレーズとのコサイン類似度の平均は表 5.1 のようになった．

表 5.1: 後側フレーズとのコサイン類似度の平均による従来手法と提案手法の比較

手法	コサイン類似度の平均
従来手法	0.759
提案手法	0.773

提案手法では従来手法よりもコサイン類似度の平均が 0.014 ポイント向上していることが分かる．

5.5　考察

図 5.1, 図 5.2 は従来手法と提案手法でのコサイン類似度の差が 0.411 ポイントと最も大きい曲の MIDI 楽譜データである．この曲において，従来手法の後側フレーズとのコサイン類似度は 0.417，提案手法では 0.828 となっている．

図 5.1 は従来手法によって生成された補間フレーズで接続した曲の MIDI 楽譜データである．縦軸は音階，横軸は小節数となっている．2 本の赤線の間が生成されたフレーズであり，主に黄枠で囲まれている部分の音 $E\flat 4$, $E\flat 5$, $E\flat 6$, $E\flat 7$ が生成されていることがわかる．これらの音は前側フレーズにおいて多用されており，それに続く形で生成されていると考えられる．このように従来手法は前側フレーズを考慮したフレーズの生成が行えることは確認できたが，後側フレーズの考慮はできないため，補間フレーズから後側フレーズに入った途端，使用される音高が突然切り替わったようになっている．

図 5.2 は提案手法によって生成された補間フレーズで接続した曲の MIDI 楽譜データである．縦軸は音階，横軸は小節数となっている．従来手法と同様に前側フレーズから続く形で $E\flat 4$, $E\flat 5$, $E\flat 6$, $E\flat 7$ が生成されているが，従来手法ではあまり見られなかった緑枠部分の音も多く生成されている．これは前側フレーズだけでなく後側フレーズも考慮できるようになったことで，後側フレーズで使われている音を参照して生成が行われているためだと考えられる．そのため，提案手法では前後のフレーズをスムーズに接続できる補間フレーズの生成が可能となっている．

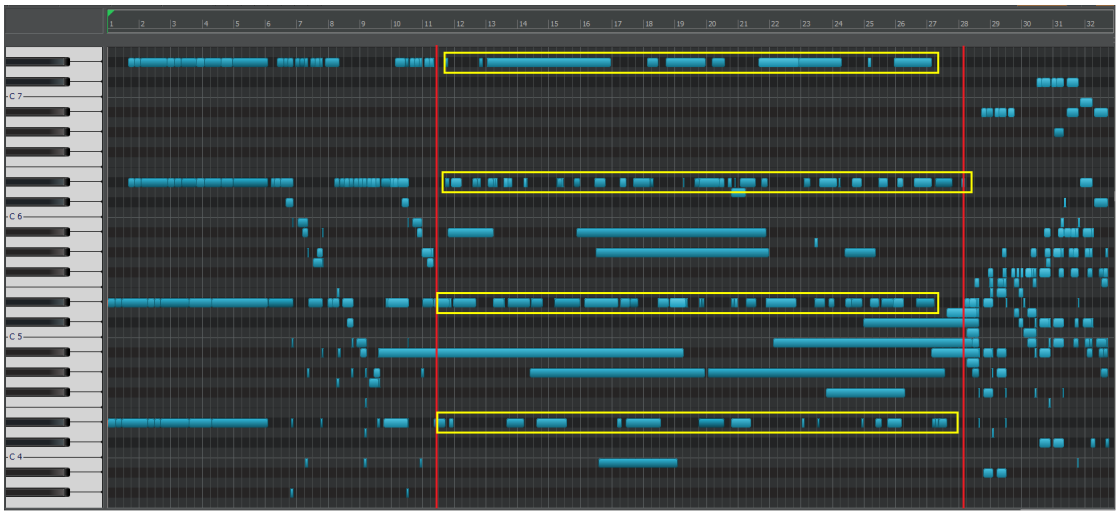

図 5.1: 従来手法の MIDI 楽譜データ

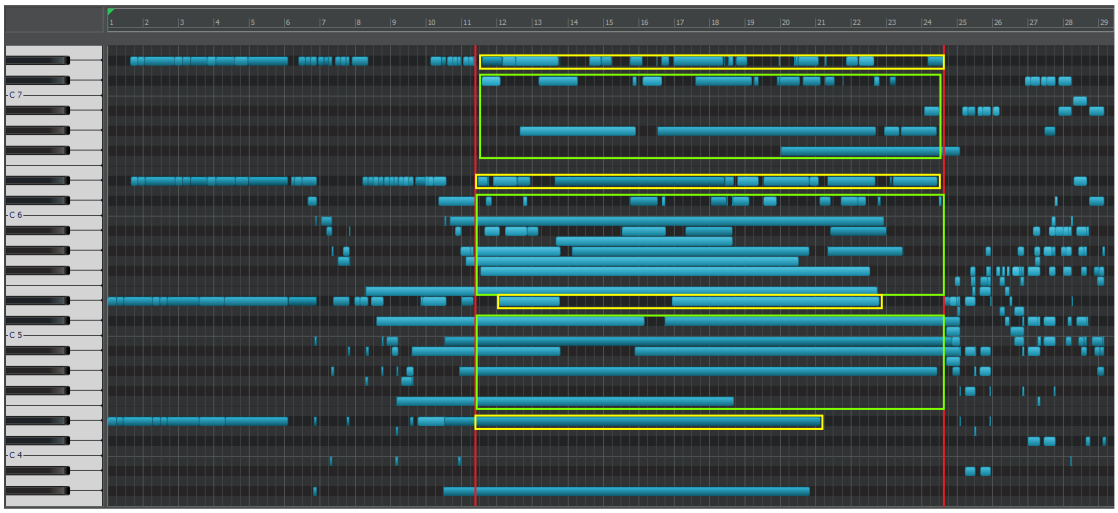

図 5.2: 提案手法の MIDI 楽譜データ

次に，コサイン類似度の差が 0.24 ポイントである図 5.3, 図 5.4 の曲について考察する．図 5.4 では，図 5.2 と同様に従来手法では見られなかった緑枠部分の音の生成が見られる．また，図 5.3 では見られた黄枠部分の音が生成されていないことがわかる．これは前側フレーズだけでなく後側フレーズも参照できるようになった結果，後側フレーズにはない黄枠部分の音の生成確率が低くなり，生成されなくなったと考えられる．

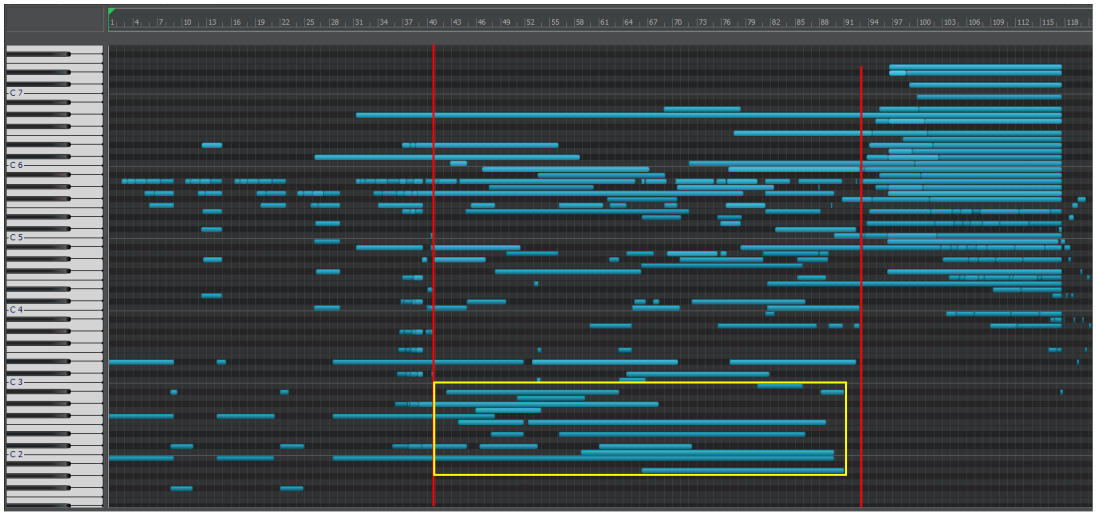

図 5.3: 従来手法の MIDI 楽譜データ

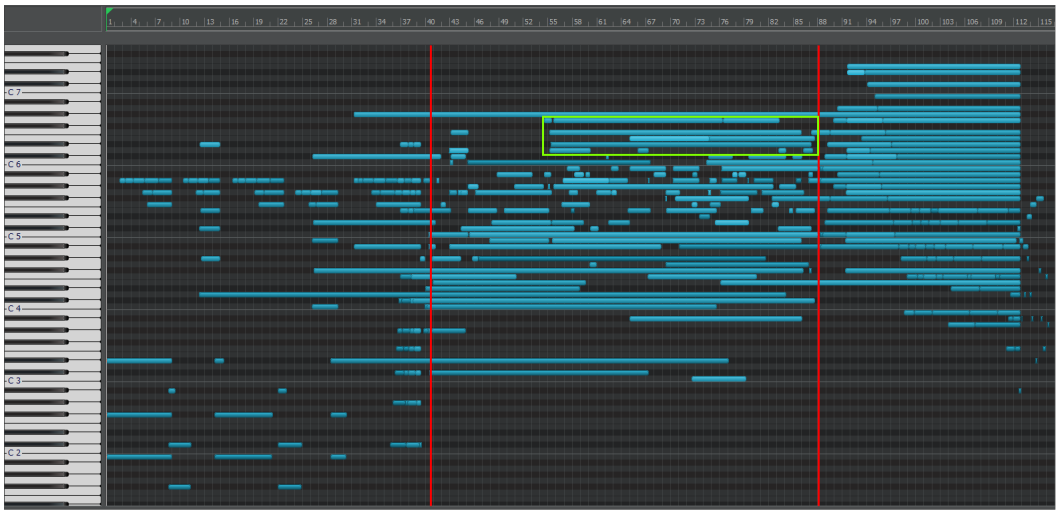

図 5.4: 提案手法の MIDI 楽譜データ

これらの考察から，提案手法では本研究の目的である前後フレーズを考慮した補間フレーズの生成を実現できていると考えられる．しかし，提案手法は従来手法と比較して不

協和音が多く生成されるという問題がある．これは，従来手法では主に前側フレーズで使用されている音・和音を用いたフレーズの生成が行われているのに対し，提案手法では前側フレーズを参照して生成する音・和音と同時に後側フレーズで使われている音・和音も生成されているため，従来手法と比較して音の構成が複雑になり，不協和音となっていると考えられる．これは，生成するイベント位置から一定距離以上のイベントを参照できないようにすることで，補間フレーズの始め付近は前側フレーズで用いられている音・和音，終わり付近は後側フレーズで用いられている音・和音が生成されやすくなり，より自然な音の推移を行う補間フレーズを生成できるようになるのではないかと考えられる．

第6章　結論

本研究では，Music Transformer[4] を用いて，フレーズ間の補間を行う手法を提案した．従来手法は前側フレーズに続く形で逐次的に次のフレーズを生成するという手法であり，後側フレーズを考慮した生成ができないという問題があった．そこで本研究では，マスク付き Self-Attention のマスク形状を変更し，後側フレーズを考慮した上で前側フレーズの続きを逐次的に生成するという手法をとることにより，従来手法の問題解決を図った．生成された補間フレーズと後側フレーズのクロマベクトルに対する実験により，前後フレーズを考慮することで，提案手法は従来手法よりコサイン類似度の平均が 0.014 ポイント向上することを確かめた．

提案手法は前側フレーズだけでなく後側フレーズで用いられている音・和音も参照して補間フレーズを生成するために多様な音高が生成されるが，それにより従来手法と比較して不協和音が多く生成されるという問題がある．今後は生成するイベント位置から一定距離以上のイベントを参照できないようにするという手法を導入することにより，提案手法の問題を改善することが考えられる．

謝辞

本研究にあたり，大小様々な場面でご協力していただいた全ての方々に，感謝とお礼を申し上げます．指導教員として日頃から長い時間ご指導賜りました B4 グループ獅子堀正幹教授，大野将樹講師に心から感謝申し上げます．

構築したシステムに対する意見や助言，研究室内の備品管理など，その他多くの場面においてお世話になりました B4 グループ獅子堀研究室内の先生先輩方に深く感謝申し上げます．

同級生として屈託のない意見などを伝えてくれた獅子堀研究室内部学部 4 年生一同に心から感謝の意を表します．

参考文献

[1] 清水柚里奈, 菅野沙也, 伊藤貴之, 嵯峨山茂樹, “動画解析・印象推定による動画 BGM の自動生成,” 研究報告音楽情報科学 (MUS) 2015.17 (2015): 1-6.

[2] Ian Simon, Dan Morris, and Sumit Basu, “MySong: automatic accompaniment generation for vocal melodies,” Proceedings of the SIGCHI conference on human factors in computing systems. 2008.

[3] 安藤 大地, 伊庭 斉志, “対話型 GP を用いたクラシック音楽のための作曲支援システム,” 芸術科学会論文誌 4.2 (2005): 77-86.

[4] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Ian Simon, Curtis Hawthorne, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck, “Music Transformer: Generating Music with Long-Term Structure,” International Conference on Learning Representations (2019).

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin, “Attention is all you need,” Advances in neural information processing systems 30 (2017).

[6] Sageev Oore, Ian Simon, Sander Dieleman, Douglas Eck, and Karen Simonyan, “This time with feeling: learning expressive musical performance,” Neural Computing and Applications (2018): 1-13.

[7] Adam Roberts, Jesse Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck, “A hierarchical latent vector model for learning long-term structure in music,” International conference on machine learning. PMLR, 2018.

[8] 清水 光希, “Music Transformer に基づく中間フレーズの生成手法” 令和 3 年度 卒業論文

[9] Geoffrey E. Hinton, and Ruslan R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” science 313.5786 (2006): 504-507.

[10] Diederik P. Kingma, and Max Welling, “Auto-Encoding Variational Bayes,” CoRR abs/1312.6114 (2013): n. pag.

[11] Sepp Hochreiter, and Jürgen Schmidhuber, “Long short-term memory,” Neural computation 9.8 (1997): 1735-1780.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, “Deep residual learning for image recognition,” Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[13] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton, “Layer Normalization,” ArXiv abs/1607.06450 (2016): n. pag.

[14] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani, “Self-Attention with Relative Position Representations,” North American Chapter of the Association for Computational Linguistics (2018).

[15] Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer, “Generating Wikipedia by Summarizing Long Sequences,” ArXiv abs/1801.10198 (2018): n. pag.

[16] David Peer, Sebastian Stabinger, Stefan Engl, and Antonio Rodríguez-Sánchez, “Greedy-layer pruning: Speeding up transformer models for natural language processing,” Pattern Recognition Letters 157 (2022): 76-82.

[17] Emiru Tsunoo, Yosuke Kashiwagi, and Shinji Watanabe, “Streaming transformer asr with blockwise synchronous beam search,” 2021 IEEE Spoken Language Technology Workshop (SLT). IEEE, 2021.

[18] Piano-e-Competition dataset. https://www.piano-e-competition.com/