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

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の補間に適用した MusicVAE と，自然言語処理モデルの Transformer を音符系列の生成に適用した手法である Music Transformer について解説する．

2.1　MusicVAE

MusicVAE の前に，まず変分オートエンコーダの重要な点を説明する．

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．オートエンコーダの学習は，教師データとして入力データそのものを与える教師なし学習であり，次式の平均二乗誤差 (MSE：Mean Squared Error) を最小化することで行われる．

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

2.1.2　変分オートエンコーダ

変分オートエンコーダ (VAE：Variational Auto-Encoder) は，オートエンコーダと非常によく似ているが，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 $X \sim p\left(X \mid z\right)$ 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 の学習目的関数は，周辺尤度 $p(X)$ の証拠下限 (ELBO) を最大化するもので，次式が成り立つ．

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

この損失関数をオートエンコーダの損失関数と比較すると，VAE において事後分布 $q(z \mid X)$ と尤度 $p(X \mid z)$ はそれぞれエンコーダとデコーダに対応することから，第 1 項はオートエンコーダの損失に似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 $\epsilon \sim N\left(0, I\right), z = \mu + \sigma \cdot \epsilon$ 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

MusicVAE のエンコーダとデコーダには長・短期記憶 (LSTM：Long short-term memory) 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，一連の隠れ状態を生成する．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 $\mu = f_1\left(h_T\right), \sigma = f_2\left(h_T\right)$ 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) シーケンス行列で表現され，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だけに圧縮するため，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を発見している．そこで，この研究では階層型デコーダを導入している．これは，各小節に対して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．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，潜在分布のより良い学習につながる．

2.1.4　生成

2 つの異なるメロディ 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．したがって，学習した潜在分布は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．

2.1.5　問題点

MusicVAE は同時に複数の音が鳴るようなメロディを生成することができないという問題点がある．

2.2 Music Transformer

第3章　Music Transformer

Music Transformer は，自然言語処理モデルである Transformer 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．エンコーダでは，入力されたフレーズの音符系列から中間表現を獲得する．エンコーダで獲得した中間表現を線形変換することで，入力されたフレーズに続く新たなフレーズの音符系列を予測し，出力する．

このモデルは，音符系列の最初から順に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．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．

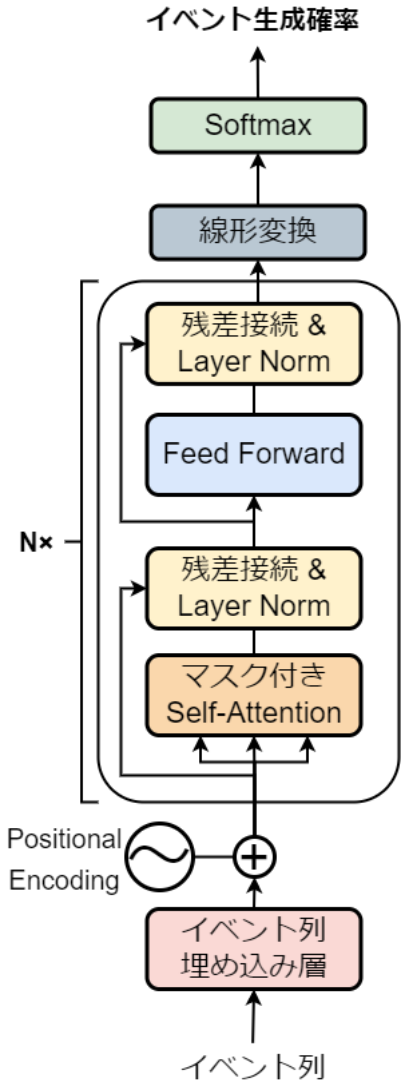

図 3.1: Music Transformer の概要図

3.1　データ表現

Music Transformer では，音符系列をベクトルに変更するため，Oore ら [2] によって提案されたパフォーマンスエンコーディングを用いる．これは，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ている．(図 3.2 参照)

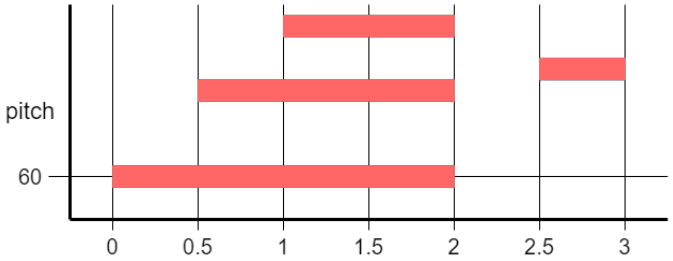

図 3.2: 左図はピアノ演奏を視覚化したもの，右図は行われるイベント (上から下に実行される)．C メジャーコードが C,E,G の順で 0.5 秒おきに演奏され 2 秒まで続き，そこで音が一度なくなる．2.5 秒から F の音が 0.5 秒間演奏され，3 秒時点で全て終了する．C メジャーコードは 80 の速度，F は 100 の速度で演奏される．

3.2　入力

Music Transformer ではエンコーダへの入力の際に，埋め込み層で入力イベント系列を埋め込み表現を表す行列に変換する．その後，Positional Encoding により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，各イベントの系列における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 関数を用いて次式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．

3.3　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 [3] を行った後に Layer Normalization [4] 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．

Transformer における Self-Attention

Self-Attention の説明をするにあたり，以下，シーケンスの長さ，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：$V = XW^V$ に変換し，W^Q, W^K, W^V はそれぞれ $D \times D$ の正方行列にする．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次式により算出される．

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

Music Transformer における Self-Attention

Music Transformer における Self-Attention では，Shaw ら [5] が提案した相対位置表現を導入し，シーケンス内で 2 つの位置がどれだけ離れているかによって Attention を考慮可能になった．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 $L \times L$ 次元のロジット行列を生成し，各ヘッドの Attention の確率は次式となる．

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

この手法により，S^{rel} を計算するためのメモリフットプリント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，長いシーケンスの適用のみに制限される．

Relative Attention のメモリ効率的な実装

この手法では，中間メモリ要件を $O\left(L^2D\right)$ から $O\left(LD\right)$ に減らすことで Relative Attention の実装を改善した．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 から必要なすべての項はすでに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．この方法と，Shaw ら [5] の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．

表 3.1: 全体の相対メモリ複雑度 (中間相対埋め込み (R または E^r) ＋相対ロジット S^{rel})，$D_h = 64$ と仮定して 16GB メモリの GPU に収まる最大系列長，および 1 層 1 ヘッドのメモリ使用量を比較 (単位：MB).

実装	計算量	最大系列長	$L = 650$	$L = 2048$	$L = 3500$
Shaw[5]	$O\left(L^2D + L^2\right)$	650	$108 + 1.7$	$1100 + 16$	$3100 + 49$
Music Transformer[1]	$O\left(LD + L^2\right)$	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ことである．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) とそれ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，非常に長いシーケンスでは，ベースの Transformer の 2 次メモリ要件でさえ実用的ではない．例えば，Wikipedia や画像生成 [6] において，入力シーケンスを重複しない

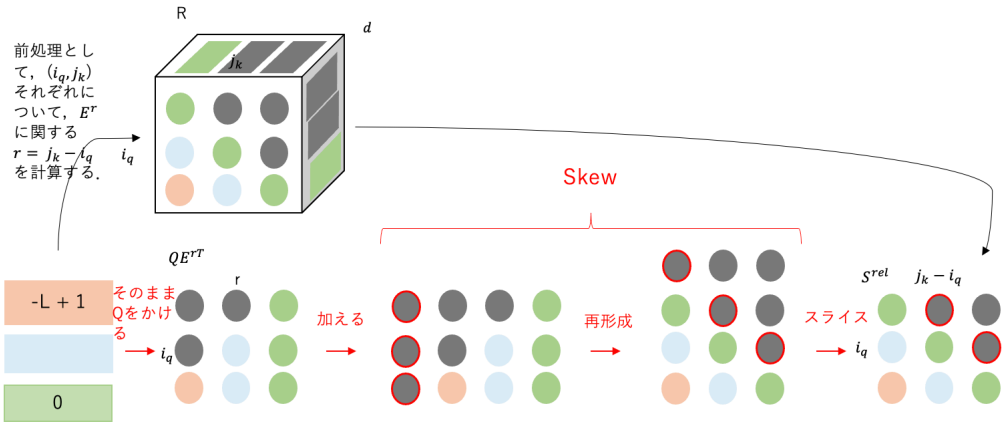

図 3.3: Relative Global Attention: 下部は skew アルゴリズムを説明するもので，R のインスタンス化を必要としない．(上段は $O\left(L^2 D\right)$ である)．灰色はマスクまたはパディングされた位置を示す．各色は相対距離の異なる位置を示す．

ブロックにチャンクすることで 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．

局所的な場合にも Relative Attention を使用するために，まず，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 $\left(\frac{L}{M}\right)^2$ と非常に小さいことに注意すること．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，以下の手順で skew 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．

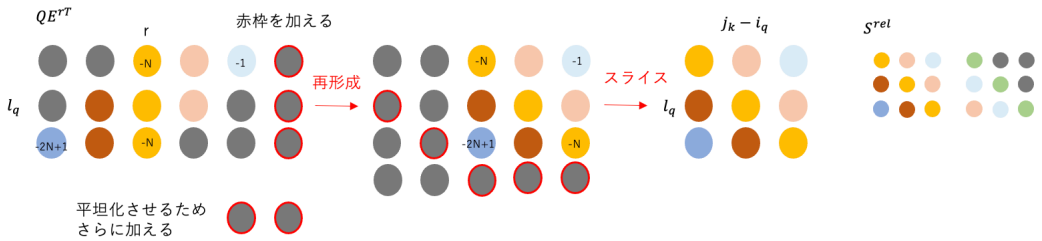

図 3.4: Relative local attention: 左図は skew の手順，右図は S^{rel} の望ましい構成を示している．

マスク付き Self-Attention

マスク付き Self-Attention では，Query，Key，Value にエンコーダの内部状態系列を用いる．ただ，推論時は，予測するイベントよりも後で生成されるイベントを知ることはできない．したがって，前方のエンコーダ系列の内部状態のみを用いる．また学習時でも，推論時との整合性をとるため，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．具体的には次のような変更を加える．各ヘッドの Attention 出力 $Z = (z_1, \ldots, z_L)$ の要素 z_i は次式で表せられる．

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

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) を次のように変更する．

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

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 $\alpha_{ij}\,(i < j)$ は次のように 0 となる．

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

Feed Forward サブレイヤ

各ヘッドの Attention 出力は連結，線形変換され，$L \times D$ 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．次に，Feed Forward (FF) サブレイヤは，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．

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

出力

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 関数を使用することで，予測された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．

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

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

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 [7] やビーム探索 [8] 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点がある．

第4章　提案手法

本章では，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) を次のように変更する．

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

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 $\alpha_{ij}\,(i < j \ \&\ j < L_A + L_B)$ は次のように 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を図 4.1 で示す．

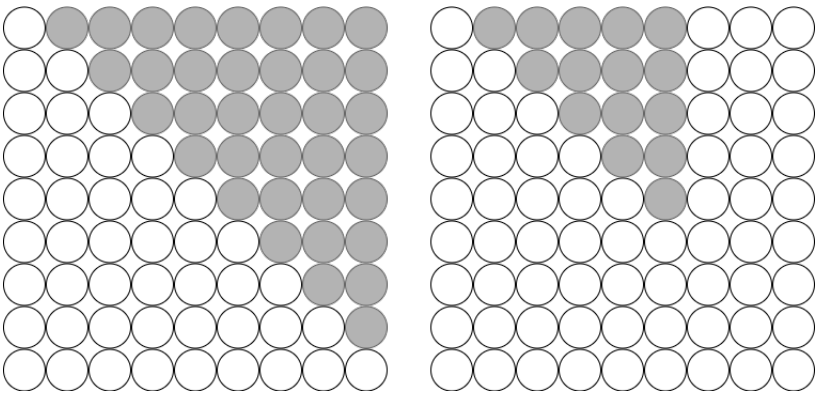

図 4.1: 左図が従来手法，右図が提案手法のマスク．灰色はマスクの位置を示す．l 行目は l つ目以前のイベントと後側 L_C つのイベントとの関連度を捉えられる状態で $(l+1)$ つ目のイベントを推測できることを表している．

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ることが可能となり，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．

例えば，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様子を図 4.2 で示す．

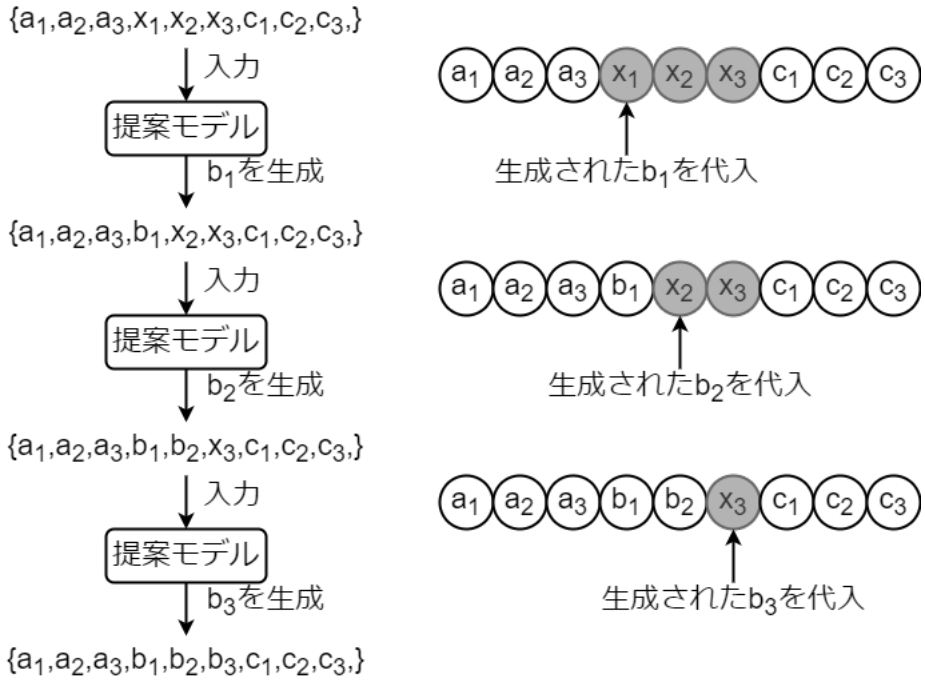

図 4.2: 左図が提案手法の補間フレーズ生成の流れ，右図は生成時のマスクの位置を灰色で示している．

第5章　実験

5.1　データセット

本実験では，データセットとして Piano-e-Competition dataset[9] を用いる．学習用データとして，Piano-e-Competition dataset より 2287 曲を用いる．また，テスト用データとして，Piano-e-Competition dataset より 254 曲を用いる．

5.2　学習及び生成

以下に実験に用いたパラメータ及び loss の推移 (図 5.1) を示す．

学習時のバッチサイズ 4

入力できる最大の音符系列の長さ 2048

層の数 6

Multi-Head Attention における Head の数 ?

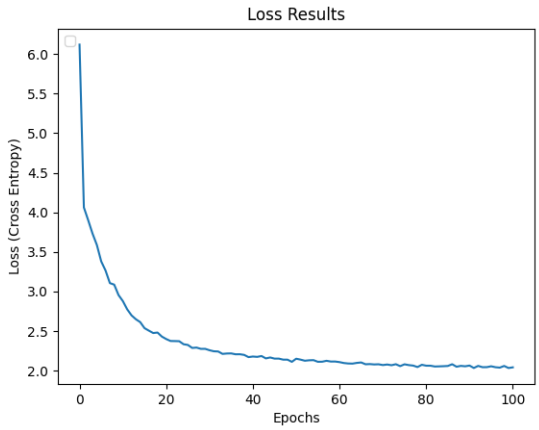

図 5.1: loss の推移 (これは清水先輩の手法の画像のまま)

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 1024 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 1024 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 1024 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

定量的評価と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中間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．式 5.1 にコサイン類似度の定義を示す．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

5.5　考察

第6章　結論

謝辞

参考文献

[1] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Ian Simon, Curtis Hawthorne, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, Douglas Eck (2018). “MUSIC TRANSFORMER : GENERATING MUSIC WITH LONG-TERM STRUCTURE,” arXiv:1809.04281v3 [cs.LG] 12 Dec 2018.

[2] Sageev Oore, Ian Simon, Sander Dieleman, Douglas Eck, Karen Simonyan (2018). “This Time with Feeling:Learning Expressive Musical Performance,” arXiv:1808.03715, 2018.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun (2015). “Deep Residual Learning for Image Recognition,” arXiv:1512.03385v1 [cs.CV] 10 Dec 2015.

[4] Jimmy Lei Ba, Jamie Ryan Kiros, Geoffrey E. Hinton (2016). “Layer Normalization,” arXiv:1607.06450v1 [stat.ML] 21 Jul 2016.

[5] Peter Shaw, Jakob Uszkoreit, Ashish Vaswani (2018). “Self-Attention with Relative Position Representations,” arXiv:1803.02155v2 [cs.CL] 12 Apr 2018.

[6] Peter J. Liu, Mohammad Saler, Etienne Pot, Ben Goodrich, Ryan Sepassi, Łukasz Kaiser, Noam Shazzer (2018). “GENERATING WIKIPEDIA BY SUMMARIZING LONG SEQUENCES,” arXiv:1801.10198v1 [cs.CL] 30 Jan 2018.

[7] David Peer, Sebastian Stabinger, Stefen Engl, Antonio Rodoríguez-Sánchez (2021). “Greedy Layer Pruning: Decreasing Inference Time of Transformer Models,” arXiv:2105.14839v1 [cs.CL] 31 May 2021.

[8] Emiru Tsunoo, Yosuke Kashiwagi, Shinji Watanabe (2020). “STREAMING TRANSFORMER ASR WITH BLOCKWISE SYNCHRONOUS BEAM SEARCH,” arXiv:2006.14941v4.

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