Vol.2018-GI-39 No.11
2018/3/2

リズムゲームの上達を支援するコンテンツ自動生成法

Liang Yubin[1,a)]　池田心[1,b)]

概要：リズムゲームとは，リズムや音楽に合わせてプレイヤがアクションをとることで進行する形式のゲームである．多くの場合，リズムゲームのコンテンツ（求められるアクションとタイミング）は，音楽素材から人間デザイナーが作成しているため、素材は無数にあっても遊べるコンテンツの数は限られている．本研究では，ディープラーニングを主な手法として，音楽素材からコンテンツを自動的に生成することを目的として研究を行う．我々は音声データを入力，コンテンツを出力とする教師あり学習を行い，難易度がさまざま・正例が少ないといった学習データの特性に対する工夫を行う．さらに，プレイヤの上達を支援するために，難易度自動調整や，苦手なアクションパターンの検出・挿入などの工夫を提案する．

キーワード：リズムゲーム，自動生成，上達支援，難易度自動調整

Procedural Contents Generation of Rhythm Games for Self-Training

Yubin Liang[1,a)] Kokolo Ikeda[1,b)]

Abstract: Rhythm game is a game form which the player takes actions in accordance with rhythm and music. In many cases, the contents (required action and timing) of rhythm games are generated by human designers from music materials, therefore the number of available contents is limited, while there are unlimited number of materials. In this research, we aim to generate contents automatically from music materials by using deep learning methods. We implemented a supervised learning which inputs music data and outputs contents, and tried to solve some difficulties caused by characteristics of learning data, such as various levels/difficulties and very few positive labels. In addition, to support the strength improvement of players, we propose methods such as automatic adjustment of difficulty level, detection and insertion of difficult action patterns.

Keywords: rhythm game, automatic generation, strength improvement support, automatic difficulty adjustment

1. はじめに

リズムゲームとは，リズムや音楽に合わせてプレイヤがアクションをとることで進行する形式のゲームである．リズムゲームは誰にでも馴染みやすい「音楽」を主な題材として扱っており，遊び方が感覚的に解りやすく難易度も調整可能のため，広い層に人気のゲームジャンルとなっている．

多くの場合リズムゲームのコンテンツ（求められるアクションとタイミング）は，音楽素材から人間デザイナーが作成しており，素材となる音楽が無数にあったとしても，ゲーム上で遊べるコンテンツの数は限られている。さらに，プレイヤの好みの音楽のものがなかったり，仮に複数の難易度があったとしても，適した難易度がなかったりする場合も多い．

本研究では，深層学習技術を主な手法として，音楽素材からコンテンツを自動的に生成することを目的として研究を行った．我々は音声データを入力，アクションのタイミングと種類を出力とする教師あり学習を行った．機械学習問題としては，1) 学習データに用いたコンテンツの難易度がさまざまで，同じ音楽でも出力が全く異なるものがあること，2) アクションが求められるタイミング（正例）の割合が非常に低いこと，という困難さがある．これに対し，本研究では，既存研究[3]も参考に 1) 難易度を特別な入力として扱うこと，2) 曖昧ラベルを与えて正例の焼き増しを行うこと，で対応する．さらに，プレイヤの上達を支援するために，難易度自動調整や，苦手なアクションパターンの検出・挿入などの工夫を提案する．

2. 背景

2.1 「OSU!」

与えられたコンテンツのみならず，プレイヤがコンテンツを作ることもできるオープンソースのリズムゲーム「OSU!」が 2007 年に開発された[1]．現在，「OSU!」の登録アカウントは 1 千万を超え，世界中にいろんなコンテストが不定期に開催されている．

「OSU!」のゲームコンテンツは beatmap と呼ばれる単位で扱われ，音楽に対して，「どのタイミングで」「どのアクションを取るべきか」が記録されている．「OSU!」のゲーム形式はいくつかあり，本研究で想定するのはそのうちの一つ mania モード（図 1）である．このモードでは，上部から落下

1) 北陸先端科学技術大学院大学
Japan Advanced Institute of Science and Technology
a) liang_yubin@jaist.ac.jp
b) kokolo@jaist.ac.jp

図 1：「OSU!」の mania モード

する「マーカー」が「判定ライン」と重なる瞬間にアクションする（ボタンを押すなど）ことが求められる．このタイミングが適切なら高得点となり，押し間違えればミスとなる．ほとんどの場合このタイミングは音楽のリズムと同期しているため，楽曲を自分が演奏しているような臨場感を味わうことができる．

本稿では，「OSU!」の mania モードの 4K マップ（4 key）において，音楽素材から beatmap を自動的に生成する研究を行う．4K マップとは，操作するボタンが 4 つのゲームコンテンツである．ボタンの操作は，押してすぐ離すものと，“長押し”と呼ばれるものがある．長押しでは，離すタイミングも適切でなければならない．

2.2 関連研究

Jan と Bock の論文[2]では，深層学習を用いて音の始まり（musical onsets）を抽出する実験が行われた．Chris，Lipton，McAuley の論文[3]では，LSTM（Long short-term memory）[4]を用いて Dance Dance Revolution というリズムゲームのコンテンツの自動生成が行われている．LSTM は循環ニューラルネットワークユニットの 1 種で，時間上の勾配消失を有効に抑えられ，音声や動画などの時系列データに適している．しかしこれらの自動作成法では，数通りの難易度を指定できるものの，「ある（アクション組み合わせの）パターンは得意だが、あるパターンは苦手」といった“個人の癖”には対応できていない．リズムゲームのプレイヤの各々の習熟段階に合わせて上達を支援するようなコンテンツ自動生成法が求められている．

3. 提案システム

現在、多くのリズムゲームのコンテンツは、音楽素材から人間デザイナーが作成しているため、素材となる音楽が無数にあっても，ゲームの中で遊べるコンテンツの数は限られている。また，一つの曲は通常数分程度は続き，その中には個人ごとに得意なアクションパターンと苦手なアクションパターンが登場することもあるが，その苦手なものだけを練習することは通常できない．例えば，ピアノの練習であれば一曲のうち苦手な場所だけを繰り返し弾くことができるが，一般のリズムゲームでは一曲の最初から該当部位に至るまでやりなおす必要がある．それらの現状を踏まえ，リズムゲームのプレイヤの各々の習熟段階に合わせて上達を支援するようなコンテンツ自動生成法が求められていると考えた．

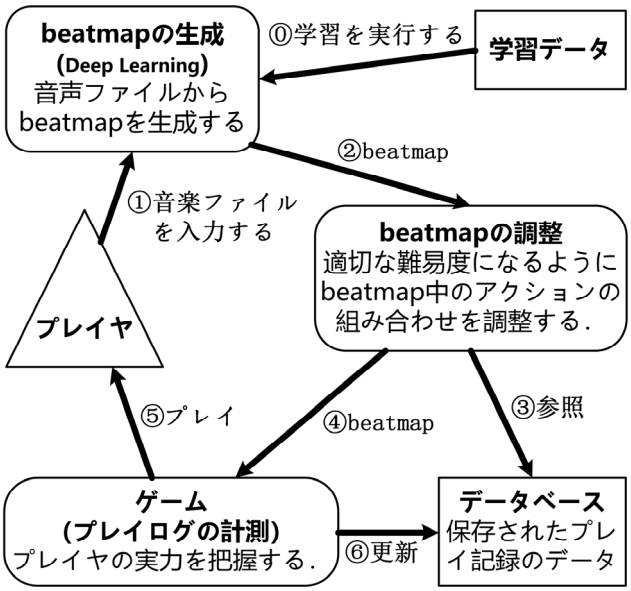

図 2 提案システムの全体フレームワークとデータの流れ

提案しようとするシステムは主に 3 つのモジュールによって構成される．「音声ファイルから beatmap を生成する」「プレイヤの実力を把握する」「適切な難易度になるように beatmap 中のアクションの組み合わせを調整する」の 3 つである．本稿執筆時点では，beatmap 生成モジュールまでが作成されている．提案システムの全体フレームワークとデータの流れは次のようにする予定である．図 2 に全体像を示し，説明は概ね図中の数字に対応している．

(0) 事前に，「beatmap の生成」モジュールのオフライン学習を実行する．これは全プレイヤ共通に行い，以降はプレイヤごと個別に行う．

(1) プレイヤが「自分の好みの音楽ファイル」と「初期難易度」をシステムに入力する．音楽ファイルは楽譜や midi など音符に分離された形式である必要はなく，wav，mp3 などの生データで良い．

(2) 生成モジュールが beatmap を生成する．この時点では各プレイヤ向けではなく，万人向けである．

(3) 調整モジュールが，各プレイヤ向けの調整を行う．データベースから過去のそのプレイヤのプレイ履歴を参照する．最初の時点ではデータベースは空もしくは意味のないほどに少ないので，調整は行わない．

(4) 調整された beatmap が，OSU!のメインシステムに与えられる．

(5) プレイヤは OSU!を通じて生成された beatmap をプレイする．

(6) そのプレイ結果はデータベースに蓄積され，そのプレイヤの実力が多角的に測定される．
(7) データが蓄積されていくと，そのプレイヤにあった難易度調整や，苦手パターンの追加が行えるようになる．

4. beatmap の生成

本章は，beatmap の生成の仕組みについて述べる．我々はこれを二段階の教師あり学習で行う．まず音声データと難易度を入力，アクションのタイミングを出力とする教師あり学習を行う．続いて，アクションのタイミングを入力とし，アクションの種類を出力とする教師あり学習を行う．二段階を通して beatmap の生成モジュールを構築する．

4.1 学習データの特徴

本稿の学習データは「OSU!」のホームページから収集した．アクセス時点まで累計 10 万回以上プレイされた（つまり人気の高い）beatmap パックを集計対象とし，学習データとして前処理を行う．一般的に，beatmap パックは音楽 1 曲と，異なる難易度の beatmap を複数含んでおり，プレイヤが同じ曲に対して，“ある程度は” 自分のレベルに合わせて選ぶことができる．学習データの統計を表 1 に示す．

データセット	
作者数	約 300 人
曲数	473
曲全持続時間	約 64000 秒
beatmap 数	1655
beatmap 全持続時間	約 215000 秒
アクション数	約 169 万
アクション数/秒	7.85

表 1 ：学習データの統計

なお，“長押し” は，押すことと放すことの２つの離れたタイミングがあるので，これらを別のアクションとしてカウントした．すなわち，各アクションは（クリック，押し，放し）の種別と，時刻を持つということである．本稿の学習データは，特定・少数のデザイナーによって作られたものではなく，アマチュアも含む世界中の多数の作者によって作られたものである．そのため，作者の個性や作風の違いは非常に多様であり，同じ曲・同じ難易度であっても全くタイミングや種類・配置が違う beatmap が提供されていることもある．これらは教師あり学習を行う際には不都合な特徴であるといえる．

4.2 beatmap の “難易度” の定義

学習データには同じ曲についても様々な「プレイヤが感じる難易度」のものが含まれている．また，我々は同じ曲について様々な難易度の beatmap を作成したい．これらの要請から，音楽データとは別に「well-defined な難易度」パラメータを入力として含めることにする．

プレイ時に「プレイヤが感じる難易度」に影響する要素は多い．その要素は概ね以下の 5 つにまとめられる，
(1) アクション組み合わせの複雑さ
(2) 単位時間内のアクションの数（密度）
(3) バー（図 1 参照）が判定ラインへ移動するスピード
(4) アクションのミス判定の厳しさ
(5) ミスが許容される回数

このうち(3)〜(5)は，コンテンツである beatmap というよりは，ゲームシステムの設定である．本稿では，(1)および(2)の項目のみに着目する．

(1)と(2)のうち，(1)は人により得手不得手がありスカラ化も困難であるため，我々は well-defined な難易度として仮に(2)密度のみを用いることにする．学習データ 215000 秒分を，密度つまり秒あたりのアクション数を 10 段階に分けたとき，各密度段階の占める時間合計のヒストグラムを図 3 に示す．

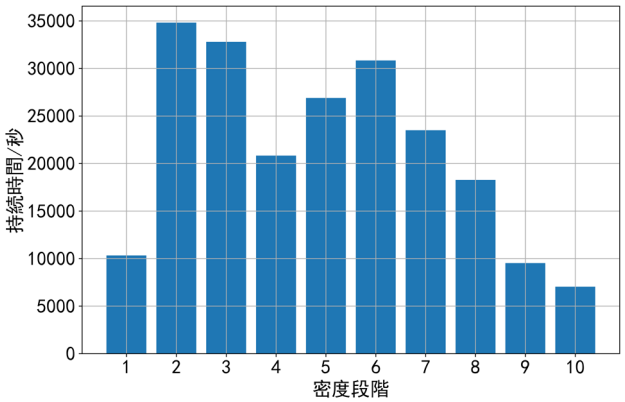

図 3 ：各密度の beatmap の合計時間

4.3 アクションのタイミング生成

4.3.1 音楽特徴量と学習ラベル

第一段階の教師あり学習では，波形データを入力としてタイミングを出力とする．本研究では過去の論文[2]の方法に基本的に従い，対象問題の違いを考慮した工夫を加える．生の波形データから，特徴量としてメル尺度[5]を抽出して用いる．抽出の帯域は 27.5Hz から 16kHz まで，80 バンドのメルフィルタバンクを用いてメル尺度を求める．3 種類の時間幅 23ms，46ms，93ms を用い，ウィンドウの移動幅すなわち 1 フレームは 10ms とする．すなわち，1 秒あたり 100 回，そこにアクションがあるかどうかを推定するわけである．音楽の連続性から，各フレームの前後 7 フレームを特徴量として用いる．すなわち，10ms ごとに，全部で 15 フレーム×80 バンド×3 ウィンドウサイズの入力があることになる．

学習ラベルは，クリック，押し，放しのタイミングを同じタイミングとし，区別しない．これらを区別したり，4 つあるボタンのうちどれを押させるか（図 1 参照）を決定したりは第二段階の教師あり学習の担当である．

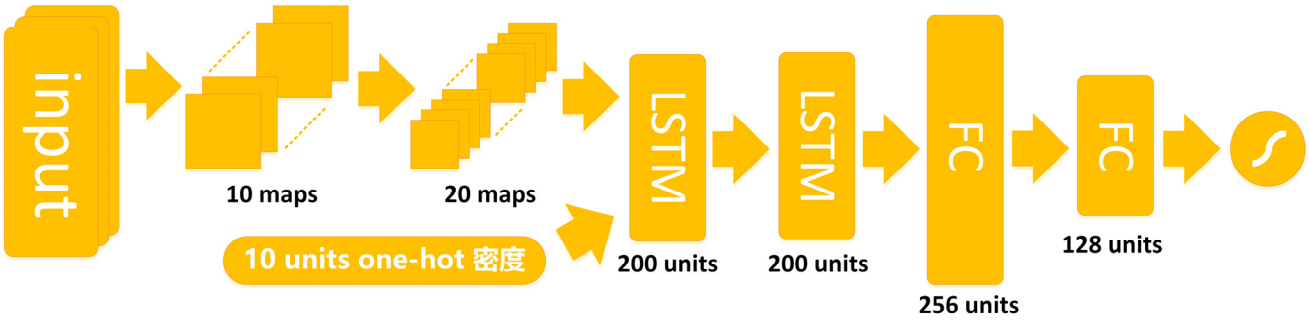

図 4 C-LSTM フレームワーク

4.3.2 タイミング予測のニューラルネットワーク

アクションのタイミング生成のニューラルネットワークは論文[3]に述べられた C-LSTM モデルである．本稿はそれに工夫を加えて双方向 LSTM[6]とし，また曖昧ラベルの工夫を用いて性能向上を図る．

C-LSTM の概念図を図 4 に示す．入力となる音楽特徴量は 15×80×3 つの 3 次元位相データである．第 1 層の畳みカーネルは 7×3×3，10 個である．第 2 層の畳みカーネルは 3×3×10，20 個である．各畳み層は Max Pooling を用いる．LSTM 層の入力に難易度（密度）を表す 10 ユニットの one-hot ベクトルを加え，ここの値を変えることで同じ音楽データであっても出力されるタイミングの難易度を調整できるようにする．畳み層の次は 200 ユニットの LSTM 層を 2 層用いる．LSTM 層の次，256 ユニットと 128 ユニットの全連結層を用いる．最後は 1 ユニットの Sigmoid 出力，該当フレームがアクションのタイミングに選ばれる確率である．畳み層と全連結層の活性化関数は ReLU，LSTM 層の活性化関数は tanh を利用する．

本稿のデータセットは，平均しても 1 秒あたりのアクション数は 7.85 個にすぎない．つまり，100 個のラベルの中，正例は 7.85 個しかない．こういった正例が少ない学習データを学習することは一般的に困難で，負例の間引きなどが用いられることもある．本研究では，正例の焼き増しにあたる曖昧ラベルを提案する．曖昧ラベルとは，図 5 に示されるように，0（そこにアクションはない）か 1（そこにアクションはある）かのような急激な変化ではなく，ある程度なだらかに RBF などのように徐々に変化するラベルである．10ms という小さいフレーム幅であれば，1 フレーム前や後ろにアクションが推定されることは“間違い”ではなくて“惜しい”というべきである．こういった状態は評価時には正解として扱うが（前研究でも同じである），それを明示的に学習に取り入れたい．このような“もっともらしさ”を曖昧ラベルは表現でき，時間順が要求されるデータにおいて適性があると考える．

もう一つ，双方向 LSTM を用いるのは，タイミング予測において，先読みする必要があるためである．既存研究は長押しというアクションが存在しないゲームを対象にしていたので過去のみを考慮に入れる片方向 LSTM でも高い性能が出たが，長押しアクションのタイミングを適切に判定するには将来の波形も考慮に入れる双方向 LSTM が有効だと考えた．

なお，過去や将来は，入力特徴量としても 7 フレーム分含まれている．しかし，予備実験によれば，70ms の入力を単に長くするよりも，双方向 LSTM を用いるほうが効率は良く，また計算量も増やさずに済むことができている．

4.3.3 アクションのタイミングの選別

ニューラルネットワークで出力されるのは，該当 10ms 長さのフレームがアクションのタイミング選ばれる確率，あるいは“もっともらしさ”である．実際に用いる場合には，ここから実際にアクションを置くかどうかを決定しなければならない．これには，閾値パラメータにより，ある確率以上ならアクションを置くことにする．閾値が高すぎればアクションは少なくなりすぎ，低すぎれば多くなりすぎる．

実際の出力例を図 6 に示す．横軸が時系列，縦軸が出力された確率である．赤点は局所的な最高点，峰である．緑線は

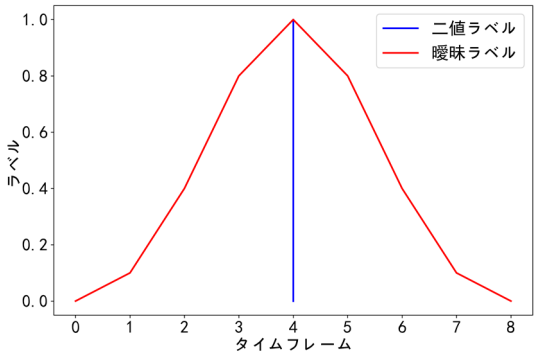

図 5 曖昧ラベル

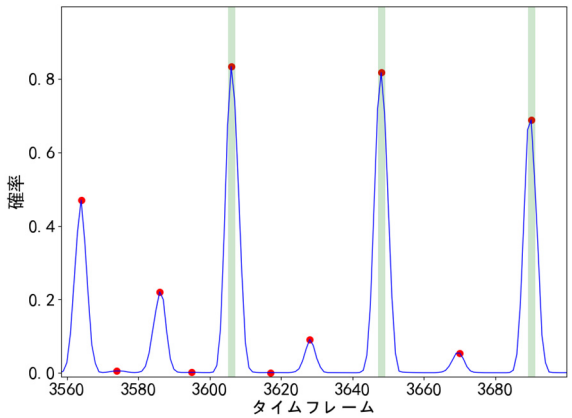

図 6 C-LSTM+曖昧ラベル+双向 LSTM の予測

学習データにおける正解（アクションがあったタイミング）である．我々は，学習時に「検証データ」を取り置き，それを用いてアクションを決定するための閾値を求めることにする．すべての検証データにおいて正しく判定することはできないので，精度と再現度のバランスをF値で表し，それが最も高くなるように閾値を求めることにする，例えば，図6の例であれば，閾値として0.6程度を使用すれば，（ここだけでの）F値は1となる．

4.4 アクションの種類の予測

第一段階で選別されたアクションのタイミングは，第二段階でアクションの種類と場所の予測に使われる．アクションのいくつかの特徴量を用いて，ニューラルネットワークを用いて予測を実現する．

4.4.1 入力特徴量

本稿では，入力特徴量として，一つ前のアクションの位置と種類（アクションコーディングと呼ぶ），前後の時間差，および指定難易度を入力にする．アクションコーディングとは，各列のアクション種類（アクションなし，クリック，長押し開始，長押し終わり）を4bitのone-hotベクトルで表記したもので，4列総計16bitのベクトルになる．時間差とは，隣接のアクションの時間上の距離である．8bitのone-hotベクトルで表記し，順で~50ms，~100ms，~200ms，~400ms，~800ms，~1600ms，~3200ms，3200ms~の8つの区間を意味する．前のアクションとの距離と，次のアクションとの距離，2つの時間差を用いて，総計16bitのベクトルになる．最後は難易度（密度）として10bitのone-hotベクトルを用いる．全部で42bitで，1が7つ立つようなベクトルとなっている．

なお，現時点では，音程は入力に含めていない．これは明らかに不自然な設定であり，多くの音楽コンテンツでは音程も考慮してデザイナーが作成していると思われる．これは今後の課題である．

4.4.2 種類予測のニューラルネットワーク

ニューラルネットワークのフレームワークは論文[3]に述べられたLSTM64モデルを利用した．入力層の次は128ユニットのLSTM層を2層用いる．出力層は256ユニットのSoftMax層である．各列で4通りの種類があるので，全部で4^4=256種類の可能アクションがあることになる． LSTM層の活性化関数はtanhを利用する．図7に概要を示す．

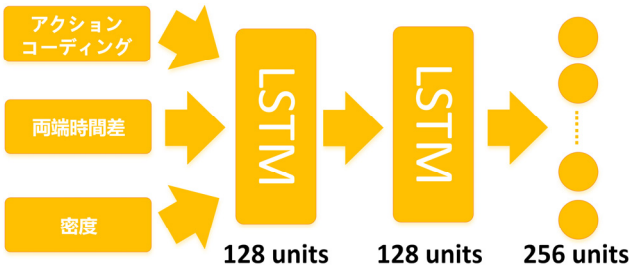

図7 アクション種類予測のフレームワーク

5. 実験

本稿では，難易度（密度）ごとのデータ数が概ね揃うように，以下の配分を行った．検証データには，各難易度ごと4曲4 beatmap，全部で40曲40 beatmapを用いた．テストデータにも同様，各難易度ごと4曲4 beatmap，全部で40曲40 beatmapを用いたが，これらは同じ音楽を使うものがないようにしてある．

学習データは，これら80曲を除いたものをベースとする．図4にあるように，難易度区分ごとに多くのbeatmapがあるものと少ないものがあるが，学習の際にはこれらがほぼ同数となることが望ましいと考えた．そこで，最も多くのbeatmapがある難易度に合わせ，足りない難易度区分では，そのデータをコピー（焼き増し）することで学習させることにした．

音声処理ライブラリはlibrosa 0.5.1，ニューラルネットワークライブラリはTensorFlow 1.2を使用した．実験の評価にあたり，我々は各手法のテストデータにおけるmicro F-scoreを用いる．

5.1 タイミング予測の実験条件

ニューラルネットワークの学習において，我々はTruncate Backpropagationという手法を使用した．バッチサイズは256にする．入力データはウィンドウかつバンドごとにZ-Scoreを用いて標準化する．各LSTM層に50%のドロップアウトを用いる．

比較するのは，オリジナルのC-LSTM，これに双方向LSTMの工夫を加えたもの，曖昧ラベルの工夫を加えたもの，両方を加えたものの4つである．

5.2 タイミング予測の結果

図8に，検証データのF-scoreの推移を示す．各手法で25エポックまで学習し，その中で一番（検証データの）性能の良い重みを用いて，最終テストデータにおけるmicro F-scoreで評価する（表2）．

実験の結果から，曖昧ラベルと双向LSTMはそれぞれ，性能向上をもたらすことが分かった．さらに，図8から，曖昧ラベルを用いるとエポック数進んでも過学習が生じにくくな

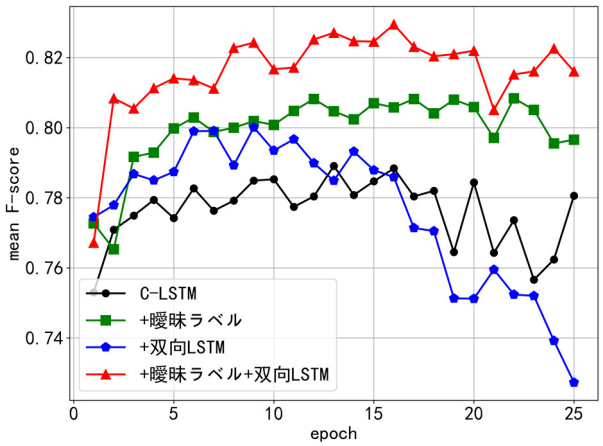

図8 アクション種類予測のフレームワーク

	Precision	Recall	F-score
C-LSTM	0.7806	0.8545	0.8159
+曖昧ラベル	0.8217	0.8381	0.8298
+双向 LSTM	0.7892	0.8440	0.8157
+曖昧ラベル+双向 LSTM	**0.8320**	**0.8543**	**0.8430**

表 2 テストデータの micro F-score

ることも分かった．逆に，双向 LSTM は，最高性能は向上するが，過学習に注意する必要があることが分かった．

5.3 アクション種類予測の実験条件

アクション種類予測では，同じく Truncate Backpropagation を用いた．バッチサイズは 128 にする．各 LSTM に 50%のドロップアウトを用いる．正則化スケールは 0.0001 にする．これらの設定はあまり丁寧に選択されたものではなく，中途の結果である．

5.4 アクション種類予測の結果

図 9 がアクション種類予測の正解率の推移を示したものである．最終的なテストデータの正確率は 0.4366 になった．実際には，不正解の中にも，「アクションの種類は合っているが場所が 1 つ違う」などの惜しいものもあり，この 43.66%と

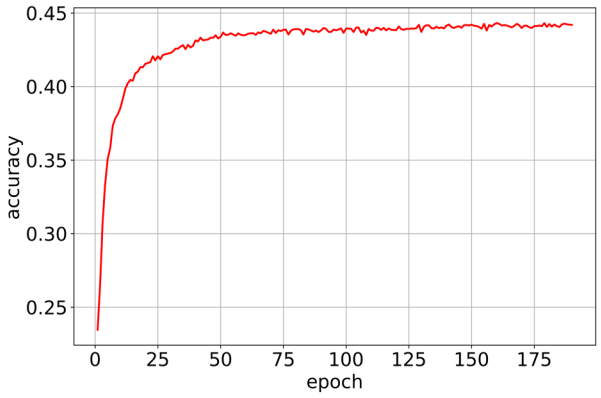

図 9 アクション種類予測の学習曲線

いう正解率はさほど小さいとは言えない．

しかし，実際にプレイしてみると，実際に出力されたアクションは特定のパターンを繰り返すループに陥りがちであった．これは，常に一番出力値の大きいアクション種類を SoftMax で選んだ結果であった．

これを解決するため，ルーレット選択を導入することにした．256 種類の可能アクションを出力値の高い順に並べ，順位ごとに定めた確率でランダムに選択することを試みた．例えば，1 位を 0.4，2 位を 0.24，3 位を 0.144 などの確率で選択する．このような工夫により実際にプレイしてみたところ，ある程度のレベルで音楽にあった楽しい beatmap が生成されうることが伺われた．被験者実験はまだこれからである．

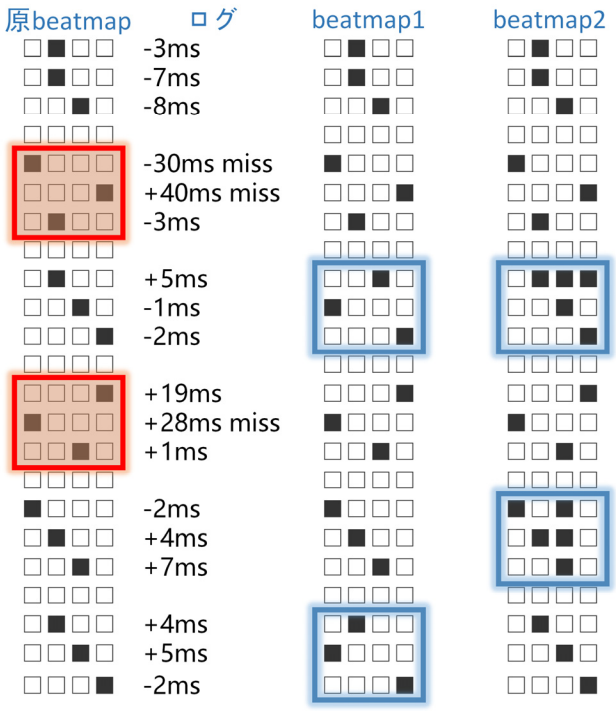

図 10 上達支援の概念図

6. 上達支援

従来のリズムゲームのコンテンツの「ゲーム上の難易度」は，曲単位で決められている．「プレイヤにとっての難易度」に影響する要素は様々だが，プレイヤが自分でそれらを細かく決めることはできない．例えば，簡単だと分類されるコンテンツの中に，あるプレイヤにとって不得手な部分が存在することがあったとする．その部分をプレイヤは繰り返し練習したいが，そのためには最初から遊ぶ必要があり，場合によっては他のほとんどの部分はそのプレイヤにはつまらないものになる．そういった不適切，不便なところに着目し，我々は「アクション組み合わせ」を自動調整する試みを行う．

6.1 アクション組み合わせ

beatmap はある時刻でのアクションを表すが，そこには前後関係があり，いくつかの連続したアクションがセットとして意味を持つことがある．これを我々は「アクション組み合わせ」と呼ぶ．

図 10 左端は，beatmap の例である．ここでは求められるタイミングの情報は削除され，どのボタンを押す必要があるかだけが示されている．この中で，例えば一番下の 3 行は，右から 3 つのボタンを 1 つずつ押していくパターンを表しており，そのパターンは中ほどでももう一度繰り返されている．

6.2 プレイヤ実力の把握

プレイヤの実力を把握するためにはプレイログを記録して利用することができる．要求されるタイミングと，実際にそのアクションが行われたタイミングを比較すれば，プレイヤがそのアクションおよびアクション組み合わせが，得意であ

るか不得意であるかを判定することができる．

図 10「ログ」の部分に，その誤差の例を表示している．一番下の 3 行およびそれが繰り返された部分について見てみると，その誤差は最大 5ms であり，このパターンはこのプレイヤにとって難しくないことが分かる．

一方で，赤枠で囲った部分は，誤差が大きく，ミスと判定されるなど，苦手としていることがわかる．

6.3 beatmap の調整

プレイを繰り返していれば，どのパターンが得意で，どのパターンは（偶然ではなくて）苦手としているのかが分かるようになる．このログに基づき，例えば以下のように調整をすることが，上達を支援するために有効であると考える．具体的なカリキュラムはまだ考えていないが，ユーザがどれかを自由に選べるようにするだけでも，今のシステムよりは良いと考える．

図 10 beatmap1：苦手な組み合わせを多く練習できるようにしたい場合，beatmap 1 青枠で示したように，得意な部分を苦手な部分で置き換えるようにして調整することができる．まったく同じパターンだけでなく，曲に合わせるなどして少し変えたものを加えてもよい．

図 10 beatmap2：簡単なセクションがつまらない場合，beatmap2 青枠で示したようにこれを難しくする調整が考えられる．

実際のシステムの運用において，実力に難易度を合わせて得手なアクション組合せをますます磨く機能とか，徐々に難易度を上げるように挑戦的なレベルとなるコンテンツも自動生成できる機能とか，一曲の中に苦手なアクションを多めにさり気なく盛り込む練習用機能とか，色々な上達させる機能も実装していきたい．

7. 結論と今後の課題

我々は，深層学習を主な手法として，音楽素材からコンテンツを自動的に生成することを目的として研究を行った．既存研究で用いていたゲームよりも利用の便利な OSU!を用い，その際に現れた課題に対処した．難易度がさまざま・正例が少ないといった学習データの特性に対し，曖昧ラベルを提案し，その有用性を実験で実証した．双向 LSTM による性能の向上も実証した．

さらに，プレイヤの上達を支援するために，難易度自動調整や，苦手なアクションパターンの検出・挿入などの工夫を提案した．

しかし，まだ不足な点が色々残っている．例えば，現時点のアクション決定モジュールは，音楽の特徴を入力特徴量としていない．あるいは，タイミング決定モジュールも，音楽の特徴（クラシックか，ポップか，ジャズか，歌かなど）を考慮したものにすべきである．これらを解決し，またできるだけ楽しく有効に訓練するための支援技術の開発実装もしていきたい．

参考文献

[1] OSU! home page: https://osu.ppy.sh/　アクセス時間：2017 年 6 月
[2] Schluter, Jan, and Sebastian Bock. "Improved musical onset detection with convolutional neural networks." Acoustics, speech and signal processing (icassp), 2014 ieee international conference on. IEEE, 2014.
[3] Donahue, Chris, Zachary C. Lipton, and Julian McAuley. "Dance Dance Convolution." arXiv preprint arXiv:1703.06891 (2017).
[4] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.
[5] Stevens, Stanley Smith, John Volkmann, and Edwin B. Newman. "A scale for the measurement of the psychological magnitude pitch." The Journal of the Acoustical Society of America 8.3 (1937): 185-190.
[6] Schuster, Mike, and Kuldip K. Paliwal. "Bidirectional recurrent neural networks." IEEE Transactions on Signal Processing 45.11 (1997): 2673-2681.