Dance Dance Gradation: A Generation of Fine-Tuned Dance Charts

Yudai Tsujino, Ryosuke Yamanishi

Dance Dance Gradation: a generation of fine-tuned dance charts

Yudai Tsujino and Ryosuke Yamanishi

Ritsumeikan University, Japan,
{is0221rs@ed, ryama@fc}.ritsumei.ac.jp

Abstract. This paper proposes a system to automatically generate dance charts with fine-tuned difficulty levels: Dance Dance Gradation (DDG). The system learns the relationships between difficult and easy charts based on the deep neural network using a dataset of dance charts with different difficulty levels as the training data. The difficulty chart automatically would be adapted to easier charts through the learned model. As mixing multiple difficulty levels for the training data, the generated charts should have each characteristic of difficulty level. The user can obtain the charts with intermediate difficulty level between two different levels. Through the objective evaluation and the discussions for the output results, it was suggested that the proposed system generated the charts with each characteristic of the difficulty level in the training dataset.

Keywords: Rhythm-based video games · Procedural content generation · Difficulty adjustment

1 Introduction

There are so many genres of video games which are the popular entertainment in the world. Rhythm-based video games are one of the popular game genres. In most of the rhythm-based video games, players perform some actions corresponding to the displayed chart. Dance games, such as *Dance Dance Revolution*, are typical and popular rhythm-based video games all over the world. Playing dance games has attracted attention not only as a lot of entertainment but also fitness conditioning[1].

There are varied types of players, beginners to experts, for the rhythm-based video games. So, multiple charts for each difficulty level are prepared for the same song. The game creators manually compose these multiple charts in general. The difficulty level of the charts is discretely composed, thus there is sometimes too much distance between the difficulty levels. Some of the players, especially middle class players, are not enough satisfied with the charts because the charts in the appropriate difficulty level for the player are not prepared. For example,

[1] https://www.engadget.com/2006/12/21/west-virginia-university-study-says-ddr-helps-fitness-attitude/ (Retrieved on Mar. 20, 2018.)

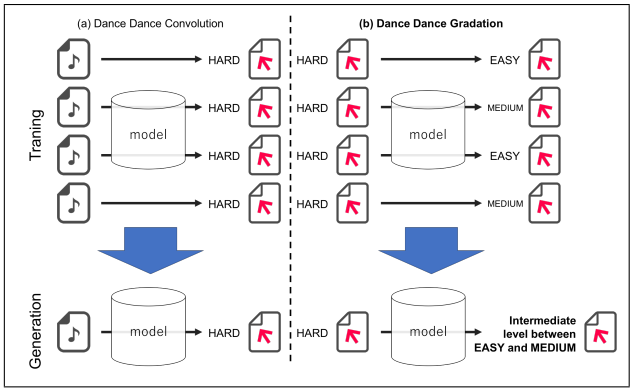

Fig. 1. The comparison of Dance Dance Convolution: the existing system (a) and Dance Dance Gradation: the proposed system (b). File icons with arrow show the charts with the corresponding difficulty level.

if the easiest chart is too easy but the second-easiest chart is too difficult for a player, the player might require the intermediate difficulty level between those. In order to comply with such requests, the game creators have to compose an indefinitely large number of charts: that is impossible by the manual. We believe that automatic generation of fine-tuned charts has been demanded. *Dance Dance Convolution (DDC)* [2] is the system to compose dance charts automatically from audio tracks. In DDC, the big task of generating dance charts is divided into two subtasks; step-placement and step-selection tasks. DDC generates dance charts based on the relationships between acoustic features and the charts. The user can specify the difficulty level of the charts to be generated on five-level because the model in DDC has the one-hot vector for difficulty levels as the input. However, more detail tuning of difficulty levels cannot be supported in DDC. Moreover, it has been reported that the quality of the easier charts generated by DDC is not enough.

This paper proposes *Dance Dance Gradation (DDG)* that is a fine-tuning system for difficulty levels of dance charts. For the step-placement task, by blending the training dataset in different difficulty levels, DDG fine-tunes the difficulty levels of the generated charts. Fig. 1 shows the concepts of DDC and DDG. Machine learning acquires the relation model between input and output in the training dataset. That is to say, the output of the learned model should have the averaged characteristics of the training dataset. As inputting multiple difficulty levels as the training dataset, the generated chart should be the averaged difficulty level based on the mixing ration of difficulty levels. Then, we propose the method to determine the threshold for onset detection considering the mixing ration.

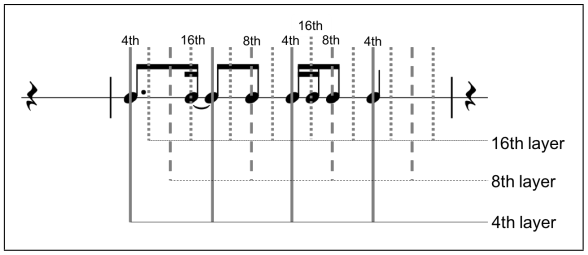

Fig. 2. The concept of the beat layer. The note should belong to the lowest layer. For example, the last note in this figure is assumed as the step in the 4th layer, though it also belongs to the 8th and 16th layers.

1.1 Definition

Some definitions key to this paper are as follows. A set of timing obtained by dividing a bar into n equal parts ($n \geqq 4$) is defined as "nth beat layer." Fig. 2 shows the idea of the beat layer and the corresponding musical score. Let the lowest layer in the layers which the timing where a given step exists belongs to be the lth layer, the step should belong to lth layer and defined as "lth step" or "the step in the lth layer"; here, "the lowest" means least n. This expression is defined as "beat layer" in this paper. The "4th" and "8th" above the notes in Fig. 2 shows the beat layer of each note. This expression does not strictly equal to a quarter note but is common in rhythm-based video games. This custom might arise from the note length not being taken into the consideration for playing the rhythm-based games.

If some steps belong to the higher layer, the sequence of steps should be more difficult and complex. Conversely, the sequence of steps belonging to the lower layer is easy to be sensed. Difficult charts have steps belong to the higher layer and easy charts consist of the steps belonging to lower layers.

In this paper, we used ITG dataset that is also used in the existing paper [2] [2]. Table 1 shows statistics of the dataset. The multiple charts for different difficulty levels for a single audio track are contained in the dataset. Each chart is named as "Beginner (B)," "Easy (E)," "Medium (M)," "Hard (H)," and "Challenge (C)" in ascending order of difficulty.

1.2 Contribution

This paper offers the following contributions;

– We propose the system for constructing dataset with data which have different characteristics for supervised learning. Training with this dataset, the

[2] "In The Groove" `http://stepmaniaonline.net/downloads/packs/In\%20The\%20Groove\%201.zip` and "In The Groove 2" `http://stepmaniaonline.net/downloads/packs/In\%20The\%20Groove\%202.zip`
(Retrieved on Nov. 28, 2017.)

Table 1. The statistics of the dataset (with reference to the existing paper by Donahue *et al.* [2]).

Dataset	ITG
Num authors	8
Num packs	2
Num songs	133
Num charts	652
Publishd year	2004∼2005

learned model generates new data which reflects the strong point of each characteristic.
– We set the metric to evaluate how well-balanced output reflecting each characteristic. In this paper, we use this metric for determining the threshold to binarize the probability of step-placement.

2 Related Work

Procedural content generation (PCG) is a field of research for generating game contents automatically [7]. In PCG researches, the game is automatically generated depending on player's skill and behavior. Hastings *et al.* proposed *Galactic Arms Race* in which the weapon of the character is automatically evolved with player's behavior logs and preferences [3]. Pedersen *et al.* modeled the player's behaviors and proposed a method to generate suitable map for *Super Mario Bros.* [6]. This study can be categorized to one of those PCG researches with machine learning method.

Dynamic Difficulty Adjustment is a field of research for dynamically changing the difficulty levels of games based on the skills and actions of the players [4]. Andrade *et al.* resolve the problem of adjusting the actions of the computer-controlled opponent in the real-time games with reinforcement learning [1]. In this paper, we manually set the dataset for the training. If the training data would be set based on the players' skills, the difficulty levels can be dynamically and automatically adjusted. How to model the skills of the players and how to apply the model would be our future work for Dynamic Difficulty Adjustment in rhythm-based games.

This paper can be categorized not only the entertainment computing but also music information retrieval researches. In the field of music information retrieval, adjusting the difficulty levels for musical instruments has been a popular topic for a long time. For instance, Yazawa *et al.* proposed a method to generate guitar tablature adapting with player's level from the audio track [8] and Nakamura and Sagayama proposed a method to reduce piano score with merged-output hidden Markov model [5]. Though such reduction methods would be powerful for some particular instruments, different rules and features should be prepared for each instrument. On the other hand, a method based on machine learning

with difficult and easy charts can be applied to other instruments if we collect huge corresponding easy and difficult charts. For actual musical instruments, it is a hard task to collect a lot of scores in different difficulty level for the same song. However, in the field of the rhythm-based games, that is the target of this paper, we can easily obtain the adequate amount of charts in different difficulty levels for the same song. The proposed system captures this specific characteristic of the rhythm-based games.

3 The Proposed System

This paper proposes ***Dance Dance Gradation*** **(DDG)**. DDG is a system with training dataset in multiple difficulty levels to learn fine-tuned difficulty levels of dance charts for the step-placement task. Only charts with a unified difficulty level are used as the training data, the model should generate charts for that level. On the other hand, using the charts with multiple difficulty levels as the training data, the model should generate the charts moderately reflecting each characteristic of the difficulty level stored in the dataset. For example, if both Easy and Medium charts are used as the training data in 50% and 50%, the generated charts would be harder than Easy but easier than Medium. That is, DDG **gradationally** tunes the difficulty level of dance charts by arranging the training dataset.

Note, we use the same step-selection model as DDC with no modification, the chart is generated through step-placement and step-selection tasks, though. The details of step-selection model can be found in the existing paper [2].

3.1 Constructing dataset for training characteristic of each difficulty level

In dance games, the characteristic of charts is different depending on difficulty levels. Through the objective analysis for ITG dataset, we confirmed that the following score features related to the step-placement task are different depending on difficulty levels;

Feature 1: Frequency of steps The more the number of steps the higher the difficulty level is; Beginner charts have approximately 0.6 steps per second on average while Challenge have approximately 4.9 steps per second.

Feature 2: Rhythm complexity The rhythm in the easy chart is more likely to be more simple than the difficult chart; over 95% of steps in Beginner and Easy charts exist on 4th layer. 33% of steps in Challenge charts belong to 8th and 22% belong to 16th, where steps in easier charts hardly belong to.

Feature 3: Distribution in a bar Steps in easier charts exist at some specified positions in a bar while steps in harder charts have a wide distribution; approximately 74% of steps in Beginner charts are placed at the beginning of a bar, where only 11% of steps in Challenge are placed.

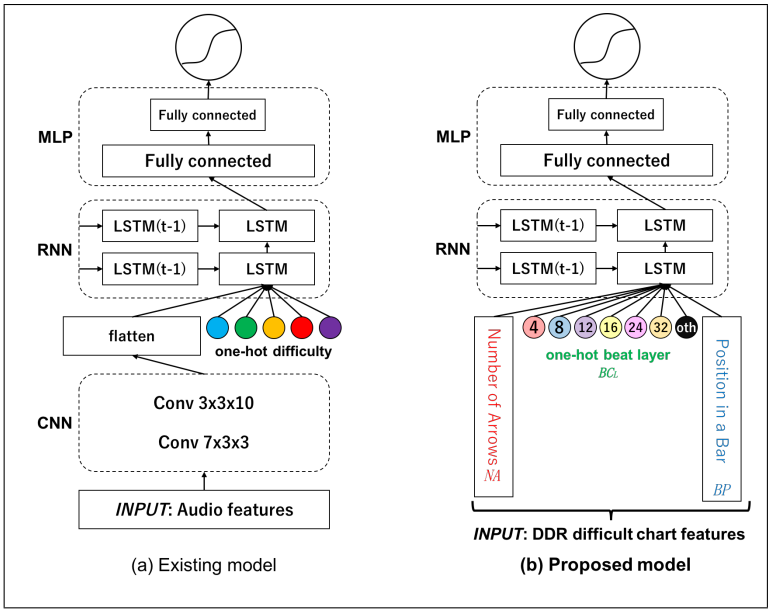

Fig. 3. Comparison of the existing model (a) and the proposed model (b).

DDG trains a model with different difficulty levels' charts. For indicating the usage rate of each difficulty level in the training dataset, we use 5-tuple vector: $\boldsymbol{DR}$. Each element in $\boldsymbol{DR}$ shows the usage rate of charts for each difficulty level in the training dataset. For example, $\boldsymbol{DR} = (1, 0.5, 0, 0, 0)$ with Challenge charts as input means that the model learns the relation of Challenge – Beginner with all tracks in the dataset and Challenge – Easy with the half of the tracks that are randomly selected from the dataset. Note, the model sometimes learns the relationships between the different output with the entirely same input.

3.2 Model

We use an LSTM model to adapt the difficulty level of charts while improving the model for step-placement in DDC [2]. Fig. 3 shows the comparative image of the models for step-placement in DDC and this paper. In DDC, acoustic features obtained from the audio track with CNN and one-hot difficulty vector are used as the input for LSTM which estimates the probability that a step is placed in the chart. The proposed model uses the score features of the difficult chart at each time as the input instead of acoustic features and difficulty vector. The following three features are used as the score features based on the **features** mentioned in Section 3.1.

NA**: Number of arrows** The number of arrows that are the required action for players at the time: non-negative integer in $[0, 1, 2, 3, 4]$. This feature is

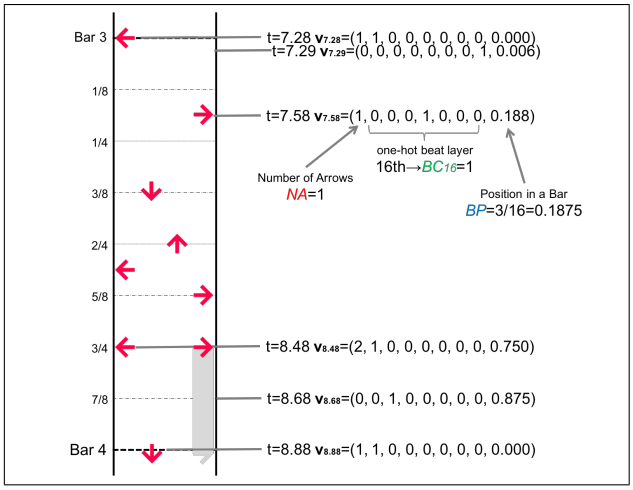

Fig. 4. Example of the score features. Score features can be expressed as 9-tuple vector for each time. The vector consists of features concerning the number of arrows, the beat layer and the position in a bar.

based on the idea that the probability of step would be low in the easy chart at the timing when the step does not exist in the difficult chart.

BC_L**: Beat layer** The 7-tuple feature that is composed of the beat layer that the time belongs to (L: 4, 8, 12, 16, 24, 32, other). That is a one-hot vector in which the corresponding element would be 1. This feature is based on the **feature 2**.

BP**: Position in a bar** The head and termination of the bar where a given time belongs to are each defined as 0 and 1, respectively. BP is the continuous value in $[0, 1)$. This feature is based on **feature 3**.

These features are connected as the input for the model. The input vector at time t: $\boldsymbol{v_t}$ is expressed as the following 9-tuple vector;

$$\begin{aligned}\boldsymbol{v_t} = (&NA, BC_4, BC_8, BC_{12}, BC_{16},\\ &BC_{24}, BC_{32}, BC_{other}, BP).\end{aligned} \tag{1}$$

Fig. 4 shows an example of the score features. We take $t = 7.58$ for a concrete example to explain the idea of the score features. There is only a right arrow at the time: $NA = 1$. To represent the time the note belongs to, the bar should be divided into 16 equal parts: $BC_{16} = 1$, thus the vector for beat layer should be $(0, 0, 0, 1, 0, 0, 0)$. The t exists at the position $\frac{3}{16} = 0.1875$ in the bar 3: $BP = 0.188$. The input vector at $t = 7.58$ would be $\boldsymbol{v_{7.58}} = (1, 0, 0, 0, 1, 0, 0, 0, 0.188)$.

The output layer is the sigmoid function, and the output would be continuous value in the range $(0, 1)$. The target at each time is expressed in binary; a step exists at the time or not. The output value concern the probability that a step is placed at the time t: $SP(t)$. Estimating $SP(t)$ for all t in a given audio track, sequential data SP that is a sequence of step probabilities can be obtained.

3.3 Setting threshold for step-placement: especially for blended difficulty level

The set of the timing for step-placement can be obtained from the series of the probability for steps SP detailed in section 3.2. The number of steps dynamically influences the characteristics of charts as **Feature 1** in section 3.1. Accordingly, the threshold for the step-placement should be appropriately determined.

The proposed system learns multiple difficulty levels, thus it is expected that the output charts should have the characteristics of all of the learned difficulty levels. Then, the characteristics of each difficulty level should be reflected in the output chart depending on the usage rate of each difficulty level $\boldsymbol{DR}$. For example, we expect that the output charts should have the characteristics of Beginner and Easy as fifty-fifty if $\boldsymbol{DR} = (1, 1, 0, 0, 0)$, and the characteristics of the output charts should be mainly Beginner-like but a little bit a flavor of Medium if $\boldsymbol{DR} = (1, 0, 0.2, 0, 0)$. Based on this idea, we evaluate;

Criteria 1 How much the output charts have the characteristics of each difficulty level used as the training data
Criteria 2 How much the balance of **Criteria 1** should be similar to $\boldsymbol{DR}$

We use F-score as **Criteria 1** which can be calculated from the comparison between the ground truth and the output charts. As **Criteria 2**, we use the harmonic mean of F-score for each difficulty level that is weighted with $\boldsymbol{DR}$. For all of the tracks in the validation data, the proposed system generates the charts while changing the threshold. And, the final threshold would be determined as the value that shows the best harmonic mean of F-score. The *threshold* is determined by the following procedures;

1. SPs for all of the tracks in the validation data are obtained.
2. All of the local maximum values are selected from SPs as the set of local maximum values.
3. The set of local maximum values is sorted in descending order of the value.
4. The threshold is determined by the stepwise approach as follows;
   for $n = 1$ *to* (*size of the set of local maximum*)
   i). The nth local maximum is used as $threshold_n$, the local maximums over than $threshold_n$ are detected.
   ii). Comparing the detected local maximums and the correct step-placement in each difficulty level, $\{F\}_n = \{F\text{-score}_B, F\text{-score}_E, F\text{-score}_M, F\text{-score}_H, F\text{-score}_C\}$ is calculated.
   iii). Calculate the weighted harmonic mean of $\{F\}_n$ as HM. The weight is equal to $\boldsymbol{DR}$.
5. The $threshold_n$ that shows the highest HM is determined as the *threshold*.

For example, as learning a model for Beginner and Easy, let $\{\mathrm{F}\}_a$ be as follows;

$$\{F\}_a = \{0.5, 0.4, NaN, NaN, NaN\},$$

Table 2. List of the model name and $\boldsymbol{DR}$. B, E, and M each mean Beginner, Easy, and Medium. The number following the initial shows the usage rate of the training data for each difficulty level. For example, M50 shows that 50% of Medium charts in the dataset is used as the training data.

	Usage rate (%)		
Model name	Beginner	Easy	Medium
B100	100	0	0
E100	0	100	0
M100	0	0	100
B100E100	100	100	0
B100M50	100	0	50
B100M100	100	0	100
B100E100M100	100	100	100
E100M100	0	100	100

here, if $\boldsymbol{DR} = (1, 1, 0, 0, 0)$, the HM would be calculated as follows;

$$HM_{\boldsymbol{DR}=(1,1,0,0,0)} = \frac{1+1}{\frac{1}{0.5} + \frac{1}{0.4}} \approx 0.44$$

if $\boldsymbol{DR} = (1, 0.5, 0, 0, 0)$, the HM would be calculated as follows;

$$HM_{\boldsymbol{DR}=(1,0.5,0,0,0)} = \frac{1+0.5}{\frac{1}{0.5} + \frac{0.5}{0.4}} \approx 0.46$$

Like the above examples, the HM is calculated according to $\boldsymbol{DR}$.

4 Experiments

We trained the model with eight patterns of $\boldsymbol{DR}$ to verify the effectiveness of DDG. Table 2 shows the list of model name and $\boldsymbol{DR}$ used for the training. All model used Challenge charts as the input. None of the eight models used neither Hard nor Challenge charts as output; the 4th and 5th elements of $\boldsymbol{DR}$ are 0. In the ITG dataset, there are 120 tracks with five kinds of the chart and 13 tracks with four kinds of the chart. We randomly divided the tracks in the dataset as 80% (107 tracks) for the training data, 10% (13 tracks) for the validation data and 10% (13 tracks) for the test data.

4.1 Training Methodology

The training methodology was same as Donahue *et al.* except for the termination condition. The target at each frame was the ground truth value. We calculated the updates using backpropagation through time with 100 steps of unrolling. The binary cross entropy is minimized by using stochastic gradient descent.

All models were trained with batches of size 256. We applied 50% dropout following each LSTM (only in the input to output but not temporal directions) and fully connected layer. All examples before the first step in the output chart

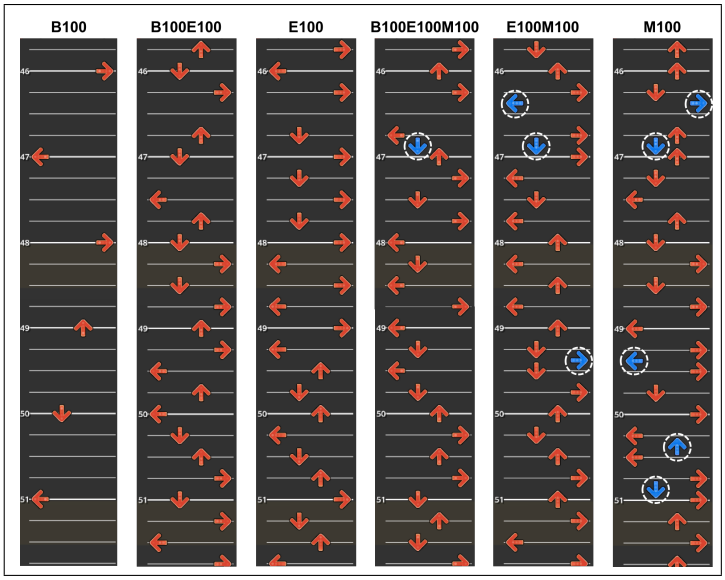

Fig. 5. Comparison of each model's prediction for the track *Queen of Light*

or after the last step were excluded. The networks were trained for 100 epochs, but the learning was cut off if the loss function did not improve through 3 epochs in the validation data; all model finished learning within 69 epochs.

4.2 Results and Discussion

Fig. 6 shows the charts for the track *Queen of Light* generated by six models. B100 model placed all steps at the beginning of the bar, and E100 model generated the sequence of 4th step. B100E100 model, which learned both Beginner and Easy charts, generated the sequence of 4th step with some quarter rests. It seemed that the chart generated by B100E100 was more difficult than the chart generated by B100 but easier than the one generated by E100. Three models using Medium generated the chart including some 8th steps (circled with dashed line). The frequency of 8th steps increased in order of B100E100M100, E100M100, and M100 model. It was suggested that the more Medium influenced the learning the harder the generated chart became.

Fig. 6 shows the charts for the track *Lemmings on the Run* generated by B100, B100M50, B100M100, and M100 model. B100 model places all steps at the beginning of the bar as same as Fig. 5. The more the rate of Medium charts is increased the more the number of steps would be placed. B100M50 and B100M100 models did not place any 8th steps though M100 model placed several 8th steps. It was suggested that B100M50 and B100M100 models learned both characteristic of Beginner and Medium charts, and learn not to place 8th steps which were only seen in the Medium charts.

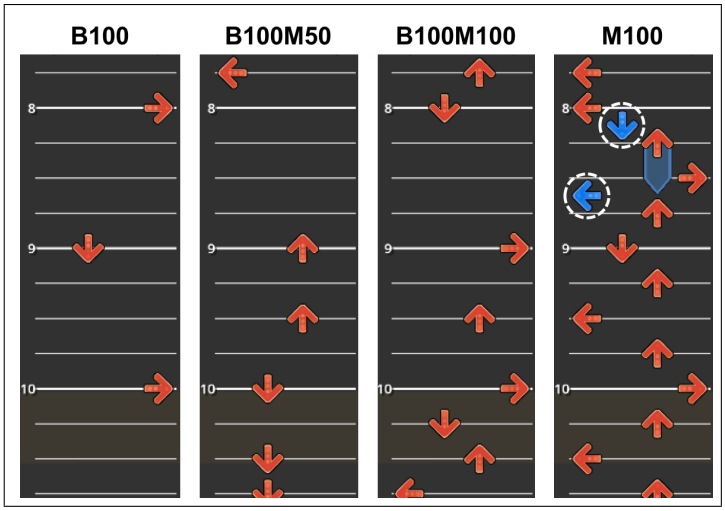

Fig. 6. Comparison of each model's prediction for the track *Lemmings on the Run*

Table 3. *F*-scores for each combination of model and answer difficulty level

	Beginner			Easy			Medium		
	F-score	Precision	Recall	*F*-score	Precision	Recall	*F*-score	Precision	Recall
B100	0.698	0.716	0.681	——	——	——	——	——	——
E100	——	——	——	0.699	0.546	0.971	——	——	——
M100	——	——	——	——	——	——	0.761	0.750	0.772
B100E100	0.408	0.270	**0.834**	0.648	0.552	0.786	——	——	——
B100M50	0.492	0.375	**0.716**	——	——	——	0.539	**0.833**	0.398
B100M100	0.436	0.300	**0.796**	——	——	——	0.649	**0.815**	0.540
B100E100M100	0.415	0.265	**0.956**	0.685	0.549	**0.912**	0.754	**0.795**	0.716
E100M100	——	——	——	0.436	0.300	**0.796**	0.649	**0.815**	0.540

4.3 *F*-score

We calculated precision, recall, and *F*-score through all tracks in the test data to evaluate the effectiveness of each model. The charts for difficulty level used in the training were used as the ground truth for those metrics. For example, we calculated the metrics against Beginner and Easy charts for B100E100 model, which used charts of those level for the training.

It is expected the model which learned multiple difficulty levels generates the intermediate level charts. It means that the generated charts should cover all steps placed in easier ground truth and have extra steps. In other words, recall is more important than precision for easier ground truth. On the other hand, these charts do not have to cover all steps in harder ground truth but should not have extra steps. For harder ground truth, precision is more important than recall.

Table 3 shows the results of the proposed models. From the results, it was suggested that B100M50, B100M100, B100E100M100, and E100M100 models satisfied those requirements; it seemed that those models could tune difficulty

level. From these results, we confirmed that DDG could generate the charts with a fine-tuned difficulty level. However, B100E100 model achieved high recall for both Beginner and Easy charts and low precision; it seemed that this model placed more steps than expected as the *threshold* for this model was set as too low.

5 Conclusion

In this paper, we present new task of fine-tuning difficulty level for rhythm-based video games. To tackle this task, we proposed a system that trains a model with multiple difficulty levels: *Dance Dance Gradation (DDG)*. For that system, we use the method to determine the threshold for nicely blending the characteristics of each learned difficulty level. Through the experiments, it was confirmed that DDG generated the appropriate charts of the averaged difficulty level reflecting on mixing ratio of learned difficulty levels.

Our future work is to use acoustic features as input combined with features of difficult charts. These features might help to generate more entertaining charts synchronized with the track.

acknowledgement

This paper was supported in part by JSPS Grant-in-Aid for Young Scientists (B) #16K21482.

References

1. Andrade, G., Ramalho, G., Santana, H.S., Corruble, V.: Challenge-sensitive action selection: an application to game balancing. In: IEEE/WIC/ACM International Conference on Intelligent Agent Technology. pp. 194–200 (2005)
2. Donahue, C., Lipton, Z.C., McAuley, J.: Dance dance convolution. In: Proc. of ICML 2017. pp. 1039–1048 (2017)
3. Hastings, E.J., Guha, R.K., Stanley, K.O.: Evolving content in the galactic arms race video game. In: Proc. of CIG09. pp. 241–248 (2009)
4. Hunicke, R.: The case for dynamic difficulty adjustment in games. In: Advances in Computer Entertainment Technology. pp. 429 – 433 (2005)
5. Nakamura, E., Sagayama, S.: Automatic piano reduction from ensemble scores based on merged-output hidden markov model. In: Proc. of ICMC2015. pp. 298–305 (2015)
6. Pedersen, C., Togelius, J., Yannakakis, G.N.: Modeling player experience for content creation. IEEE Transactions on Computational Intelligence and AI in Games **2**(1), 54–67 (2010)
7. Shaker, N., Togelius, J., Nelson, M.J.: Procedural Content Generation in Games: A Textbook and an Overview of Current Research. Springer (2016)
8. Yazawa, K., Itoyama, K., Okuno, H.G.: Automatic transcription of guitar tablature from audio signals in accordance with player's proficiency. In: Proc. of ICASSP 2014. pp. 3122–3126 (2014)