Proceedings of the Fifteenth AAAI Conference on Artificial
Intelligence and Interactive Digital Entertainment (AIIDE-19)

GenerationMania: Learning to Semantically Choreograph

Zhiyu Lin,* Kyle Xiao, Mark Riedl
Georgia Institute of Technology
{zhiyulin, kylepxiao}@gatech.edu; riedl@cc.gatech.edu

Abstract

Beatmania is a rhythm action game where players must reproduce some of the sounds of a song by pressing specific controller buttons at the correct time. In this paper we investigate the use of deep neural networks to automatically create game stages—called charts—for arbitrary pieces of music. Our technique uses a multi-layer feed-forward network trained on sound sequence summary statistics to predict which sounds in the music are to be played by the player and which will play automatically. We use another neural network along with rules to determine which controls should be mapped to which sounds. We evaluated our system on the ability to reconstruct charts in a held-out test set, achieving an F_1-score that significantly beats LSTM baselines.

1 Introduction

Rhythm action games such as *Dance Dance Revolution*, *Guitar Hero*, or *Beatmania* challenge players to press keys or make dance moves in response to audio playback. The set of actions the player is required to do timed to the music is referred to as a *chart* and is presented to the player as the music plays. Some rhythm action games involve the reconstruction of music where some sound events are played when the player performs the right action at the right time and other sound events are played automatically. This includes *Guitar Hero*—the player plays some of the guitar notes—and *Beatmania*—the player acts as a DJ that must play sound events from many different instruments; the music sounds incomplete or garbled if the player skips actions or misses the timing of actions. Other games such as *Dance Dance Revolution* involve dance moves that are choreographed to the music but does not require a strict mapping between actions and sound events; the music sounds complete regardless of what the player does.

The charts in rhythm action games are typically handcrafted, limiting gameplay to songs that already have accompanying charts. Because of this limitation, active novice communities have arisen to create new music and new accompanying charts. However, chart creation is considered a difficult and time consuming task. In this paper, we explore the use of deep neural networks to automatically construct charts for arbitrary pieces of music. We target the *Beatmania IIDX* game because it has the oldest and most mature homebrew community, providing open-source game emulators and a larger corpus of charts and music than for other games.

Beatmania IIDX is a rhythm action game in which the player reconstructs the music by playing different sound events from different instruments at precise times. Some sounds from some instruments are played automatically while others must be played by the player. That is, there are "playable" and "non-playable" sound events for each piece of music. Playable sound events appear visibly in the chart to cue the player which action to invoke and have a one-to-one correspondence with an audio sample that will be heard in the music. Non-playable sound events are considered part of the chart but are not shown to the player and the audio samples play in the background. See Figure1 for an example chart. In rhythm action game terminology, a *keysound* refers to the one-to-one mapping of a segment of music to a playable (or non-playable) chart sound event.[1]

Our technique learns a model of how chart elements relate to underlying music. Training the model consists of three tasks: (1) analyzing the music sample to identify the instruments used; (2) analyzing sample charts to identify relationships between sound events and the difficulty of charts per a variety of temporal scales; and (3) learning a model to classify whether sound events are playable or non-playable. To choreograph a new chart, the model is given a new music sample and produces a specification of which sound events should be playable and which should be non-playable. A second neural network in conjunction with heuristic rules are then used to assign controls to each playable sound event.

Our work differs from other work on neural rhythm action game chart generation in three significant ways. First, other neural generative systems (e.g., *Dance Dance Convolution* (Donahue, Lipton, and McAuley 2017)) target games that do not use keysounds—actions only need to follow the tempo without necessarily expressing any relationship to the underlying music. Second, other neural generative sys-

[1]Beatmania IIDX is *keysounded* by this definition; Dance Dance Revolution is *non-keysounded* respectively.

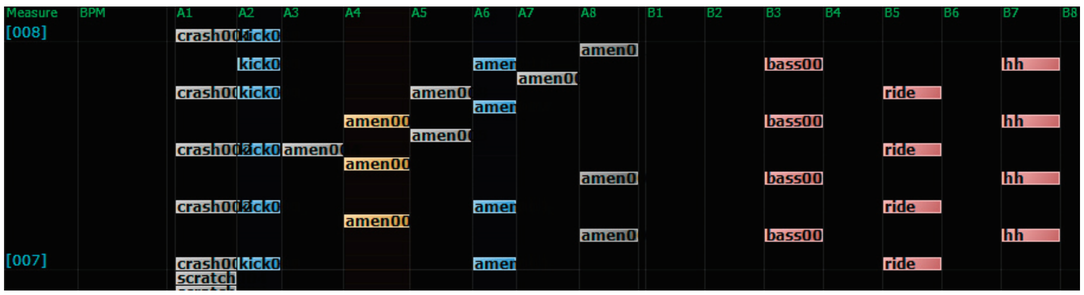


Figure 1: A visualization of a single measure of the Beatmania IIDX homebrew chart for *Poppin' Shower*. The 'A' columns are for playable sound events for different controls and 'B' columns are for non-playable sound events. The sound events are labeled with the names of the audio sample sound files. Time progresses from bottom to top, corresponding to the way sound events "fall" in the game interface.

tems use recurrent neural networks, whereas we empricially find that a specialized multilayer feed-forward network that operates on a fixed window of chart elements at different timescales works best. Third, to support novice chart creators, we explicitly designed our chart generation technique to allow users to provide target difficulty progression and bias the generator's sytlistic choices.

We conduct an evaluation to measure the precision and recall of our technique in recreating the charts in a held-out test set. Our system outperforms a number of strong baselines with respect to F_1-score, a harmonic mean of precision and recall, indicating ability to both determine when playable events should be in the chart and when non-playable events should be in the background.

In addition, we introduce the *BOF2011* dataset for Beatmania IIDX chart generation.[2]

2 Background and Related Work

The homebrew community of BMIIDX is arguably one of the oldest and most mature groups of its kind (Chan 2004), with multiple emulators, an open format (Be-music Source[3], BMS) and peer-reviewed charts published in semi-yearly proceedings[4]. Despite the community striving to provide the highest quality charts, due to the strict keysound requirements, usually the author of the music or a veteran chart author has to assist in the authoring of the chart. Many aspiring amateur content creators start by building charts for non-keysound rhythm action games (i.e. *Dance Dance Revolution* chart building is considered by the community to be easier to learn). Furthermore, there is a strong demand for customized charts: players have different skill levels and different expectations on the music-chart translation, and such charts are not always available to them.

There are a handful of research efforts in chart choreography for rhythm action games, including rule-based generation (O'Keeffe 2003) and genetic algorithms using hand-crafted fitness functions (Nogaj 2005). Procedural Content Generation via Machine Learning (PCGML) (Summerville et al. 2018) is an emerging field of study. Under its umbrella, *Dance Dance Convolution* (Donahue, Lipton, and McAuley 2017) is the first deep neural network based approach to generate rhythm action game charts. Dance Dance Convolution uses a two-stage approach. *Onset detection* is a signal analysis process that determines the salient points in an audio sample (drum beats, melody notes, etc.) where actions should be inserted into a chart. *Action selection* uses a long-short term memory neural network (Hochreiter and Schmidhuber 1997) to learn to map onsets to specific chart elements. However, while onset detection was useful for action selection, other features such as music pitch did not significantly improve accuracy. Our work on Beatmania IIDX chart generation differs from Dance Dance Convolution in that the primary challenge is determining whether each sound event should be playable or non-playable. A recent work, (Fukunaga, Ochi, and Obuchi 2019), also utilized machine learning to classify playable events. They used audio and timing features of current and 10 previous events, while we summarized these events along with other features and put them into an end-to-end pipeline.

3 Dataset

We compiled a dataset of songs and charts from the "BMS Of Fighters 2011" community-driven chart creation initiative. During this initiative, authors created original music and charts from scratch. The dataset thus contains a wide variety of music and charts and was composed by various groups of people. Most authors build 3 to 4 charts for each song. The dataset, which we refer to as "BOF2011", consists of 1,454 charts for 366 songs. Out of 4.3M total sound events, 28.7%, or 1.24M of them, are playable ones. Table 1 summarizes the dataset.

We find that modeling the difficulty of charts plays an important role in learning to choreograph new charts; this observation is also made by (Donahue, Lipton, and McAuley 2017). However, We discarded human-made difficulty labels due to its ambiguity.

Most of the charts in our dataset are relatively easy, in which non-playable sound events dominate. Furthermore,

[2]Our implementation and dataset is available at https://github.com/xxbidiao/GenerationMania

[3]http://www.charatsoft.com/develop/otogema/page/04bms/bms.htm

[4]http://www.bmsoffighters.net/

Table 1: BOF2011 dataset summary.

# Songs	366
# Charts	1,454
# Charts per song	3.97
# Unique audio samples	171,808
# Playable objects	1,242,394
# Total objects	4,320,683
Playable object %	28.7

many of the same samples are repeatedly used, such as drum samples placed at nearly every full beat throughout a chart, resulting in only 171k unique audio samples in our dataset.

4 Methods

Our chart generation system for BeatMania IIDX learns a model that relates audio samples to chart elements. There are three tasks that must be accomplished to learn this model:

1. *Sample Classification*—Identifying the instrument used in audio samples.
2. *Chart Knowledge Extraction*—Establish structure of each part in a chart including how sound events relate to each other and difficulty over time.
3. *Sample Selection*—Classifying sound events as playable and non-playable.

Tasks 1 and 2 reduce each music sample and accompanying chart to a set of sound events and summary information containing event timing, difficulty, and the pattern of instrument probabilities across different time scales (see Section 4.2). Task 3 uses the summary information to train a model that categorizes sound events as playable or non-playable.

Chart generation involves transforming a new music in the same way and predicting the playability of each sound event. Once each sound event has been categorized as playable or non-playable, a fourth task, *Note Placement*, assigns each playable sound event to a control. See Figure 2.

4.1 Sample Classification

Sample classification is a process by which notes from different instruments in the audio samples are identified. The BMS file format associates audio samples with timing information. That is, a chart is a set of sample-time pairs containing pointers to the file system where an audio file for the sample resides. Unfortunately, in the BMS file format there is no standard for how audio samples are organized or labeled. However, many authors do name their audio sample files according to common instrument names (e.g., “drums123.ogg”). The goal of sample classification is to label each sample according to the instrument based on its waveform. The predicted labels will be used to create one-hot encodings for each sample for the sample selection stage on the pipeline.

We construct a training set by gathering audio samples together with similar instrument names according to a dictionary and use the most general instrument name as the supervision label. We use the 27 most common categories for labeling, which was determined to work well experimentally. To ensure that we don’t overfit our classifier we train on an alternate dataset, “BMS of Fighters Ultimate” (BOFU) that does not share any music or charts with BOF2011, with a partially labeled dataset containing 60,714 labeled samples. Not every audio sample has a classifiable name, which we count as unlabeled.

We process the audio samples into a spectrogram representation and transform it into “audio fingerprints,” which is a vector of mel-frequency cepstral coefficients (MFCC) (Logan 2000) of the log-magnitude mel-scale frequencies over time. This mapping is done via a linear cosine transform. We also fix the bit rate of the sound to 16k, so that the representation has a consistent temporal resolution. This pipeline is shown in Figure 3.

For our model, we followed the method described in (Sainath and Parada 2015). We feed the fingerprints through two 2D convolutional layers with bias. Each of the layers is followed by a Rectified Linear Unit (ReLU) activation function followed by max-pooling. Finally, we feed the results into a fully connected layer, which then outputs a one-hot encoding of the predicted category. We use a gradient descent optimizer with 50% dropout rate and a step size of 0.01.

After training on the BOFU dataset, we achieved an 84% accuracy based on a 80-10-10% train-test-validation split.

4.2 Chart Knowledge Extraction

Inspired by community chart authoring practices and music theory we provide an abstracted blueprint of charts instead of raw chart file formats. This blueprint technically abstracts the chart from the underlying music, but references the instrument classes of each sound event, at different music timescales. This blueprint consists of three types of knowledge that is automatically extracted from a chart: *beat phase* capturing timing of sound events, *challenge model* capturing the relative difficulty sections of the chart, and a *relational summary* that captures the relationship between sound events in the chart.

Beat phase denotes the timing relationship between a sound event and its underlying music as a whole. We separate each interval representing a 4th musical note into sixteen typically 20ms windows. We assign a value between 0 and 15 for each window, 0 denoting the note being on a down beat, 8 for halfway between two down beats (8th), and so on. This yields a value for each sound event.

We find the relative difficulty of the portion of a chart that a sound event occurs in to be an important feature in determining whether a sound event should be playable or not. Our *challenge model* describes how difficult each part of a chart will be perceived by a human. We use a rule-based technique that has been used for other rhythm action games. [5] We first calculate a *strain* value for each playable event, then max pool it over 400ms intervals. The strain value of each sound event is calculated as the weighted sum of *individual strain* and *overall strain*. Individual strain is the

[5] Adapted from the Osu! framework. Their reference implementation is available at https://github.com/ppy/osu/tree/master/osu.Game.Rulesets.Mania/Difficulty

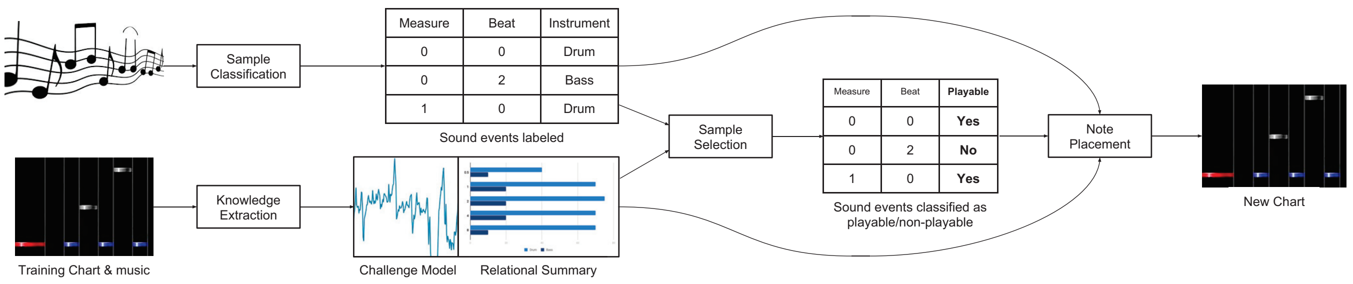
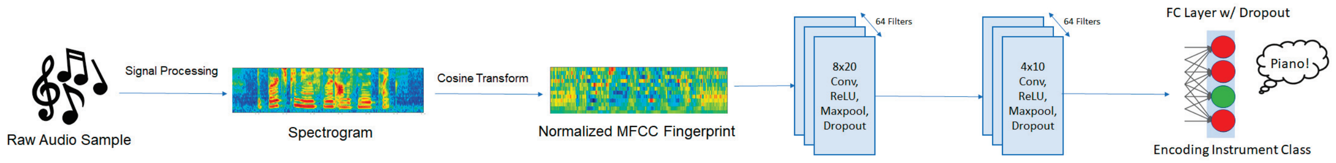


Figure 2: The chart generation pipeline.



Raw Audio Sample
Signal Processing
Spectrogram
Cosine Transform
Normalized MFCC Fingerprint
64 Filters
8x20 Conv, ReLU, Maxpool, Dropout
64 Filters
4x10 Conv, ReLU, Maxpool, Dropout
FC Layer w/ Dropout
Piano!
Encoding Instrument Class

Figure 3: The Sample Classification Pipeline.

interval between sound events mapped to the same control on an exponentially decaying scale such that short intervals have higher strain values than long intervals. Overall strain is based on the number of controls that must be activated simultaneously. We apply the pooled strain values back to each sound event, and get a value for each event.

The *relational summary* draws a big picture of the chart in different scales. Summarization is a technique popularized by *WaveNet* (Oord et al. 2016) to factor prior information from a fixed window at different time scales into a current prediction. Unlike *WaveNet* which operate with inputs at the most fine-grained level of detail, we **summarize** previous events in feature space. We found out that time scales of size 2, 4, 8, 16, and 32 beats best suit our purpose of summarizing both local and broad features. For each discrete point in time in each timescale we compute the probability that a sound event resides in this scale of each instrument class which is described in section 4.1. We illustrate relational summaries in Figure 4. Considering a sound event at time $t = 0$, we compute the likelihood that that sound event will be playable or non-playable when considering different possible sample classes and looking back over a window of time of varying time scales. Thus, each sound event is represented by an $S \times C \times 2$ matrix where S is the number of time scales and C is the number of sample classes.

For each chart the beat phase information, challenge model, and relational summary forms a *summary matrix*.

4.3 Sample Selection

Sample selection is the task of determining which sound events in the music should be playable and which should be non-playable. Our Sample Selection prediction model is a feed-forward network consisting of 4 fully connected layers of dimensions 64, 32, 16, and 2, each followed by a Rectified Linear Unit (ReLU). To perform sample selection, we pick the output node with the highest activation corresponding

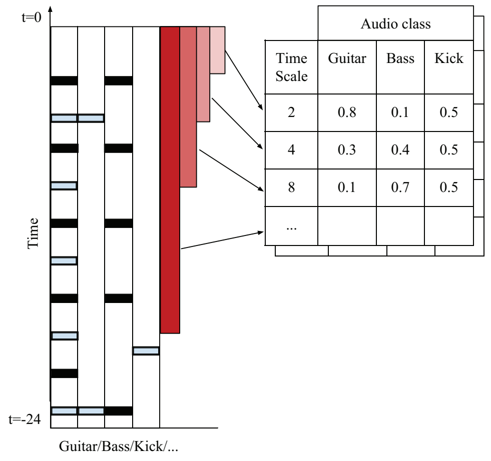


Figure 4: Depiction of a relational summary for a sound event at time $t = 0$. Black bars denote playable sound events and blue bars denote non-playable ones in the chart. Red bars indicate windows of chart history at different timescales.

to the playable or non-playable classes. Due to a class imbalance because most of the sound events are non-playable, we found that a weighted Mean Squared Error loss function helps improve the performance of the training.

At training time the summary matrix is derived from the training data so that the model can be trained to reconstruct the input data. At generation time, a summary matrix must be provided. When generating charts for new music, the challenge model and relational summary can be taken from other charts. The relational summary acts similar to a chart authoring *style*, providing an indication of preference for certain types of sound events to be made playable. For example, the relational summary from a chart with a lot of playable guitar events will bias our system toward the same. The chart from which to take challenge models and relational summaries do not necessary need to be in the dataset. It is even possible to hand-author challenge models and relational summaries to induce a specific difficulty profile and style.

4.4 Note Placement

Note placement occurs during chart generation, once we have classified each sound event as playable or non-playable. Each *playable* sound event at each timestep must be mapped to one of 8 controls.

Any process that doesn't map objects to the same control at the same time is sufficient to make a chart playable, thus note placement is not a significant contribution in this paper. We use a feed forward neural network similar to that in Section 4.3 except the final layer is a softmax prediction over the 8 controls. We post-process the note-placement with a heuristic, developed based on common chart design patterns. If more than two sound events are mapped to the same control at nearly the same time, we instead place it in mirrored position (i.e. Left-most become right-most, etc.) If that position is still occupied, we shift it to right-hand-side adjacent control. We repeat the adjacent control shift until an available position is found. Very rarely, if all 8 position is occupied, we label this sound event as unplayable and discard this event from the note placement task.

5 Experiments

We evaluate variations of our sample selection model against a number of baselines. We focus on playable/non-playable classification because any reasonable note placement will work from the player's perspective. We used a supervised evaluation metric: we embed a Summary Matrix extracted from the ground truth chart and then measure how accurate sample selection is compared to the original chart. This is a necessarily artificial test in the sense that we have a ground-truth when comparing charts back to those in the testing set.

We establish two guidelines for a good generation model: it should not only predict *playables* when they should be presented to players (high recall), but also ensure that playable events presented to players are actually playable events (high precision). A model achieving higher performance level on these two metrics demonstrates better ability to pick up essential sound events representing understanding of music resembling human author choices. We applied an 80%, 10%, 10% split on training, validation and testing data. Since the charts for the same music share similar traits, we ensured that such charts are not both in the training split and the testing split. We trained all models using the training split and report all results on the testing split.

We experiment with the following models:

- **Random Baseline:** classifies a given object as a playable with a probability of 0.3, chosen to give the best result;
- **All-Playable Baseline:** classifies all objects as a playable.
- **LSTM Baseline:** a seq2seq model (Sutskever, Vinyals, and Le 2014) with hidden and output layer size of 2. The highest activated output is selected as the prediction.
- **Full Model:** The sample selection model with summary matrix.
- **Self-Summary Model:** the same as the full model except that an empty summary matrix is initially provided and generated row-by-row on the fly based on previous generated predictions.

The LSTM baseline was chosen because the conventional wisdom, established by Donahue et al. (2017), is that an LSTM is needed to model chart progression. However, it is impossible to directly compare our model to the approach used in Donahue et al. because of the difference between keysound-based and non-keysound-based rhythm action games.

In our experiments, we explore different combinations of input features drawn from the summary matrix. We developed our Neural Network models in PyTorch. We used a mini-batch of 128 for the feed forward model; Due to the need for processing very long sequences with high temporal resolution, the LSTM model is trained by each sequence and is run in CPU mode. The Full model converges in around 2 hours in GPU mode while the LSTM model takes far longer at around 50 hours, on a single machine using Intel i7-5820K CPU and NVIDIA GeForce 1080 GPU.

6 Results and Discussion

The experiment results are shown in Table 2. We only show configurations that beat reference baselines. Our primary measure is the F_1-score, a harmonic mean of precision and recall.

The first observation is that, in all cases, relational summary features boost performance across all models that use them. The Full Model with relational summaries performs better than an LSTM model without relational summaries. This is notable because an LSTM makes use of history and attempts to model the relationship between prior time steps and the next time step. The Full Model does not make use of history directly but has indirect access in the form of the summary statistics. We hypothesize the high variance in charts hampers LSTM making it performs worse on our task.

Adding relational summaries to the LSTM model gives an LSTM access to history and also a hierarchy of statistics at different timescales. The LSTM with relational summaries achieves a precision of 0.776 ± 0.117 but only a moderate improvement in recall. This is statistically not significantly better than the precision of the Full Model at $p > 0.025$ ($N = 145$). While this version of the LSTM arguably produced the highest precision it is at the expense of recall, meaning in practice that it produced a lot of charts where sound events are not predicted as playable where they should be (perfect recall can be achieved by classifying all sound events as playable). For a rhythm action game chart, one needs both high precision and high recall.

The Full Model with relational summaries achieves the highest F_1-score at 0.700 ± 0.158, which is our primary measure. The Full Model F_1-score is statistically higher than that of the best LSTM model at $p < 0.00001$.

In our Full Model with relational summaries, the challenge model provides a 7.9% improvement in the F_1-score.

Table 2: Results for playable classification experiments, presented in mean and standard deviation. BP, CM and RS refer to Beat Phase, Challenge Model, and Relational Summary, respectively. $N = 145$ for the test set.

Model	**F_1-score**	**Precision**	**Recall**
Reference Baselines			
Random	0.291 ± 0.089	0.335 ± 0.200	0.299 ± 0.020
All Playable	0.472 ± 0.207	0.335 ± 0.199	1.000 ± 0.000
LSTM Baselines Models			
LSTM + Audio Features + BP + CM	0.424 ± 0.154	0.767 ± 0.176	0.353 ± 0.248
LSTM + Audio Features + BP + CM + RS	0.564 ± 0.149	$\mathbf{0.776 \pm 0.117}$	0.475 ± 0.194
Feed-Forward Models			
FF + Audio Features + BP + CM	0.253 ± 0.143	0.523 ± 0.266	0.179 ± 0.113
FF + Audio Features + BP + CM (Self Summary)	0.368 ± 0.198	0.422 ± 0.213	0.392 ± 0.258
FF + Audio Features + BP + RS	0.621 ± 0.206	0.760 ± 0.110	0.568 ± 0.254
FF + Audio Features + BP + CM + RS	$\mathbf{0.700 \pm 0.158}$	0.762 ± 0.114	$\mathbf{0.662 \pm 0.193}$

As with Donaue et al. (2017), we also observe that all models varied in performance on charts with different difficulty. We analyzed the effect of chart difficulty on our best performing model—the Full Model with relational summaries. Plotting overall chart difficulty against F_1-score performance on each chart (Figure 5) reveals that the Full Model achieves consistently high F_1-scores when the chart difficulty is high but has high variability in performance when chart difficulty is low. There are fewer playable sound events in low-difficulty charts and we hypothesize that there is more variance to how authors select playable events.

The Full Model is naturally sensitive to the features. We also experimented with other features including sound event density, audio pitch, different numbers of instrument classes, and a hierarchical representation with musical measure as intermediate unit of organization. These features failed to improve performance and in some cases reduced performance. We also considered (a) an auto-encoder structure for LSTM model, which tries to auto-summarize the chart, (b) multi-layer LSTM structures more similar to that in Donahue et al. (2017), and (c) using different minimal unit of resolution, such as musical beats, in classification task. However, these models either overfit quite quickly or have unrealistic computational requirements.

As a side effect of how beat-phase information is organized in our specific task we are unable to include Δ-beat features. Δ-beat measures the number of beats between the previous and next step and was used in Donahue et al. (2017). The limitation stems from (1) several semantically unrelated notes can be placed at the exact same time and (2) notes can be placed in very short intervals (such as when representing a glissando). These issues prevent effective Δ-beat detection in granularity of single notes.

Unlike *Sample Selection*, it is non-trivial to develop a quantitative method to evaluate Note Placement. A similar metric which is used in evaluation of *Sample Selection* would make little sense since no baseline Placement information is provided for non-playable notes. Furthermore, since this metric is per-note based, it will not award reconstruction of *patterns* where groups of notes are placed in a certain shape, if the pattern is misplaced. Due to these reasons, such a metric will offset drastically from human players' perception, rendering it ineffective. This urges development of metrics for qualitative studies for Step Placement model. Aside from that, our Challenge Model technique is completely heuristic based, meaning that it is sensitive to parameter tuning. A heuristic-free approach or a model learned from player experience may help in this scenario. We leave them as future work.

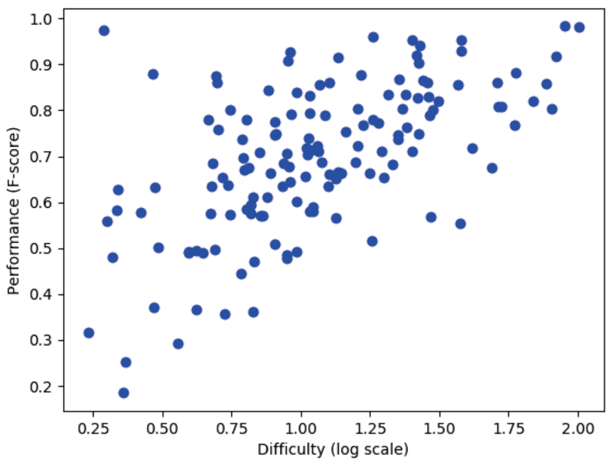


Figure 5: The performance of feed forward model (with summary) regarding relative difficulty of the ground truth chart.

7 Conclusions

Choreographing rhythm action game charts is a challenging task. Beatmania is a keysounded rhythm action game, requiring chart elements to reflect the underlying music more closely than other, non-keysounded games. We have established a technique for generating charts for new music samples that allows for user control of difficulty progression and also allows the user to bias the generator's stylistic decisions. We also provide a new chart dataset for reproducible evaluations and to facilitate further research on rhythm action game choreography.

We show that a feed-forward network with a challenge model and relational summary performs better than strong LSTM baselines at determining when playable events should be in the chart and when non-playable events should be in the background. Such a supervised evaluation is necessary to give confidence that sample selection is meaningful. The full use case is for users to provide their own challenge models and relational summary information when creating charts for new music, or to create *blends* by borrowing this information from other, existing charts and applying it to new music. Aside from solving a challenging creative task, intelligent systems that allow for user control—such as the technique described in this paper—can be of benefit to homebrew chart choreography communities by helping novices overcoming skill limitations.

References

Chan, A. 2004. CPR for the Arcade Culture.

Donahue, C.; Lipton, Z. C.; and McAuley, J. 2017. Dance Dance Convolution. In *Proceedings of the 34th International Conference on Machine Learning*.

Fukunaga, D.; Ochi, K.; and Obuchi, Y. 2019. Automatic Estimation of Key-sound in Rhythm Action Games (Retrived from https://www.art-science.org/journal/v18n1/v18n1pp10/artsci-v18n1pp10.pdf). *The Journal of the Society for Art and Science* 18(1):10–18.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Logan, B. 2000. Mel Frequency Cepstral Coefficients for Music Modeling. In *ISMIR*, volume 270, 1–11.

Nogaj, A. F. 2005. A genetic algorithm for determining optimal step patterns in Dance Dance Revolution.

O'Keeffe, K. 2003. Dancing monkeys. *Masters project* 1–66.

Oord, A. v. d.; Dieleman, S.; Zen, H.; Simonyan, K.; Vinyals, O.; Graves, A.; Kalchbrenner, N.; Senior, A.; and Kavukcuoglu, K. 2016. WaveNet: A Generative Model for Raw Audio. In *SSW*, 125.

Sainath, T. N., and Parada, C. 2015. Convolutional neural networks for small-footprint keyword spotting. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Summerville, A.; Snodgrass, S.; Guzdial, M.; Holmgård, C.; Hoover, A. K.; Isaksen, A.; Nealen, A.; and Togelius, J. 2018. Procedural content generation via machine learning (PCGML). *IEEE Transactions on Games* 10(3):257–270.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to Sequence Learning with Neural Networks. In Ghahramani, Z.; Welling, M.; Cortes, C.; Lawrence, N. D.; and Weinberger, K. Q., eds., *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc. 3104–3112.