GenéLive! Generating Rhythm Actions in Love Live!*

Atsushi Takada[1], Daichi Yamazaki[1], Yudai Yoshida[1], Nyamkhuu Ganbat[1], Takayuki Shimotomai[1], Naoki Hamada[1], Likun Liu[2], Taiga Yamamoto[2], Daisuke Sakurai[2]

[1]KLab Inc.
[2]Kyushu University
{takada-at,yamazaki-d, yoshida-yud, ganbat-n, shimotomai-t, hamada-n}@klab.com,
{liu.likun.654, yamamoto.taiga.160}@s.kyushu-u.ac.jp, d.sakurai@ieee.org

Abstract

This article presents our generative model for *rhythm action games* together with applications in business operation. Rhythm action games are video games in which the player is challenged to issue commands at the right timings during a music session. The timings are rendered in the *chart*, which consists of visual symbols, called *notes*, flying through the screen. We introduce our deep generative model, *GenéLive!*, which outperforms the state-of-the-art model by taking into account musical structures through beats and temporal scales. Thanks to its favorable performance, GenéLive! was put into operation at KLab Inc., a Japan-based video game developer, and reduced the business cost of chart generation by as much as half. The application target included the phenomenal "Love Live!," which has more than 10 million users across Asia and beyond, and is one of the few rhythm action franchises that has led the online-era of the genre. In this article, we evaluate the generative performance of GenéLive! using production datasets at KLab as well as open datasets for reproducibility, while the model continues to operate in their business. Our code and the model, tuned and trained using a supercomputer, are publicly available.

1 Introduction

The success of deep generative models is rapidly spreading over the entire fields of industry and academia. In today's game developments, deep generative models are starting to help us create various assets including graphics, sounds, character motions, conversations, landscapes, and level designs. For instance, the Game Developers Conference 2021[1] held a special session named "Machine Learning Summit" to present various deep generative models used in game products, such as for generating character motions that match the content of conversations (Ding 2021) and for generating 3D facial expression models of characters from human face pictures (Li 2021).

Figure 1: Love Live! School Idol Festival All Stars.[2]

Although deep generative models for rhythm actions have been studied for a while – notably by Donahue, Lipton, and McAuley (2017), they have been focusing on proof of concept or personal hobby use, not yet being used in cutting-edge commercial products. There thus remain questions: what is the blocker to leverage chart generation models in the game business, and how should we overcome it?

The present article reveals a key remaining problem, which is musical structure recognition. Indeed, we considered features such as beats and temporal scales with our model (see section 2.1 for more on musical structures). While existing models have not had components dedicated for these concepts, our preset model has them, which process beats and multiple temporal scales. Those correspond to our *beat guide* and *multi-scale conv-stack*, respectively.

The beat guide is an extraordinary technique in the sense that it can be computed for any input audio and straightforwardly. Somewhat surprisingly, it had been overlooked by existing works. The multi-scale conv-stack is incorporated in order to capture musical features of different time scales, like repeats and notes of various lengths.

We have included a thorough evaluation of this improved model, employing a supercomputer and user feedback from our application in the gaming industry operation with a business company. The benchmarks show that our improved model outperformed the state-of-the-art model known as the Dance Dance Convolution; DDC (Donahue, Lipton, and McAuley 2017). From the business feedback (section 6), we

*This paper is dedicated to Tomori Kusunoki and the memory of her Setsuna Yuki.

[1]https://gdconf.com/.
[2]©2013 PROJECT Lovelive! ©2017 PROJECT Lovelive! Sunshine!! ©2020 PROJECT Lovelive! Nijigasaki High School Idol Club ©Bushiroad International ©SUNRISE ©bushiroad

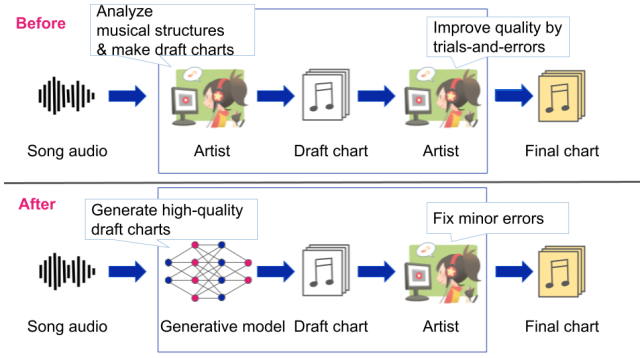


Figure 2: KLab's chart production workflow before and after this work.

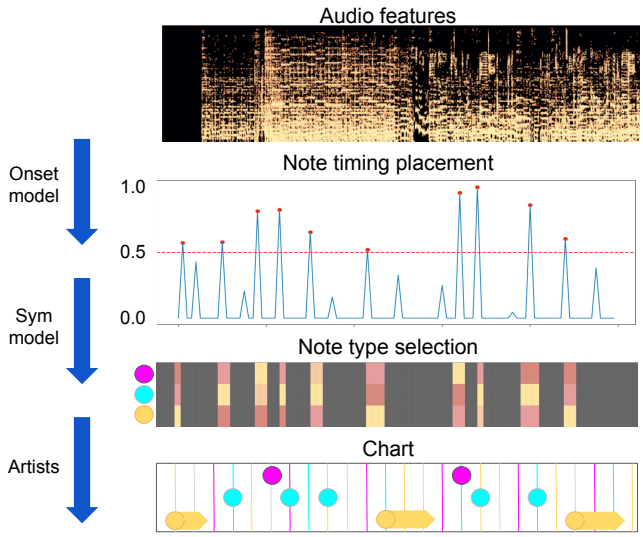


Figure 3: Data processing in the generative model. The present article focuses on the onset model.

learned that the model is capable of replacing the manual labor of generating the first drafts (this is practical, since game artists have relied on manager-level decisions to fine-tune their first drafts anyway). Our model, named *GenéLive!*, reducing the business cost by half, will stay in business operation for the foreseeable future.

To conduct the application study, we worked with the game artists team in KLab Inc., a Japan-based video game developer (which employs also some of the authors). The company has by today operated three rhythm action game titles online. A particularly successful title, "Love Live! School Idol Festival All Stars," or simply "Love Live! All Stars" (fig. 1), has been released in 6 languages and been played worldwide while under KLab's operation, acquiring more than 10 million users. Today, we see a wide range of competitive games with comparable impacts, which makes this work relevant to a larger audience.

Specifically, our target task is the generation of *charts*, which instruct the player to tap or flick buttons at specified moments, the defining challenge of rhythm action games. These buttons are known as *notes* as they fly through the screen forming a spatial pattern resembling a musical score. The audio record playing in the background is commonly referred to as a *song*.

A chart generation model consists of two submodels: the *onset*, which generates the timing of a note, and the *sym*, which decides the user action type (like a tap or flick). As deciding the timing is the bottleneck of KLab's workflow (detailed in section 2.3), this article focuses on presenting our onset model.

The present work has the following contributions:

- We propose a deep generative model for chart creation, which achieved its business-quality performance by improving the state-of-the-art model DDC (Donahue, Lipton, and McAuley 2017) by incorporating two novel techniques: the beat guide and multi-scale conv-stack.
- Each of our improvements enhances the performance for all difficulty modes in multiple game titles. The improvements were effective particularly for easier difficulty modes, overcoming a commonly known weak point of the DDC.
- Incorporated into the workflow of KLab's rhythm action titles, our model halved the chart delivery time. The workflow is usable to rhythm actions in general – the results verified the versatility also for open datasets from third parties, *Stepmania*.
- Our PyTorch-based source code and the trained models, found after extensive hyperparameter tuning (over 80,000 GPU hours of Tesla P100 on a supercomputer), are publicly available.[3]

2 Problem Definition

2.1 Musical Structure

Most songs used in rhythm action games have a typical musical structure that is composed of temporally hierarchical performance patterns. The percussion keeps a steady beat, creating a rhythmic pattern in a bar. A series of such bars, together with phrases of melody instruments or vocals, forms musical sections such as an intro, verse, bridge, chorus, and outro. For example, the song *Sweet Eyes*[4] in "Love Live! All Stars" has 60 bars that organize the following musical structure: an 8-bar intro, 16-bar verse, 10-bar bridge, 18-bar chorus, and 8-bar outro, each of which consists of repetitions of 1-bar to 4-bar phrases.

Artists at KLab confirm that this sort of musical structure is common in almost all songs in their rhythm action games and also a key feature of their chart creation. More specifically, the artists tend to put identical note patterns on the above-mentioned repetitions of a phrase. See appendix H for more details on the musical structure of Sweet Eyes and how the artists put notes on such a structure.

[3] All appendices are available on https://github.com/KLab/AAAI-23.6040.

[4] The song is available on https://youtu.be/MpAUJ36fq3g, and its portion from intro to first chorus (time range 0:16–2:07) is played in our rhythm game.

To learn the temporal patterns with our generative model, we would thus be required to consider multiple time scales. This was, in fact, absent in the network design of the DDC. To see how we designed our model to capture these features, see the explanations of the beat guide (section 4.3) and multi-scale conv-stack (section 4.4).

2.2 Game Difficulties

A song will be assigned charts of various difficulty modes ranging from *Beginner*, *Intermediate*, *Advanced*, and *Expert* to *Challenge*, in increasing order. In our preliminary experiments, the Dance Dance Convolution (DDC) (Donahue, Lipton, and McAuley 2017) generated charts for higher difficulty game modes at a human-competitive quality. (Find more related work in section 7.) However, the generation of low-difficulty charts had room for improvement (as Donahue, Lipton, and McAuley (2017) themselves pointed out). As our primary target was easier modes, this was a significant challenge.

2.3 Challenges in Business Application

Leading rhythm action titles today tend to take the form of one piece of a large entertainment franchise, like KLab's "Love Live! All Stars" does of "Love Live!." The company's role is to operate the mobile app, while songs are delivered by other participants of the franchise. After the first release of the app, KLab continued to contribute by offering new playable songs. This is why a significant cost for the company's business is posed by chart generation.

The company's workflow (fig. 2) does *not* demand a *fully*-automated chart generation, since KLab's artists need to experiment with different variations of candidate products, employing their professional skills – this is a high-level decision critical for the success of the franchise. We thus focus on semi-automation, which was to generate the *first drafts* of the charts (see fig. 3) so the artists can be freed from this low-skill labor.

To create a chart, artists repeatedly listen to the whole of a song to understand its musical structure as set by business partners. During this process, they ponder how to place hundreds of notes to be tapped by the player, to eventually craft the chart through trial and error. This first draft does not require too much expert skill, although it had been causing as much as half the cost in the workflow before our model was in operation.

The charts are then modified so that the actions are connected with emotions, like imitating the dance motion rendered in the background or flicking to a specific direction relating to the lyrics. It may be revised further to enhance the gameplay experience with more focused consideration of the overall game design.

In essence, the first draft of the chart generation is crafted only from the input audio, while the enhancements are applied from information harder to compile into numerical data. We thus target the auto-generation of the first drafts.

3 Datasets

3.1 Datasets Acquired

We acquired songs and charts used in "Love Live! School Idol Festival All Stars" (in short "Love Live! All Stars") and "Utano Princesama Shining Live" ("Utapri") operated by KLab. Both the songs and charts are provided by multiple artists. In addition, we use openly accessible songs and charts from "Fraxtile" and "In the groove" in the open source game "Stepmania," which were used also in the prior work (Donahue, Lipton, and McAuley 2017). The number of songs were 163, 140, 90, and 133 for "Love Live! All Stars," "Utapri," "Fraxtile," and "In the groove," respectively. There were typically 4 game difficulties for "Love Live! All Stars" and 5 for the rest, each difficulty contributing to one chart. See appendix A for details on the datasets.

3.2 Data Augmentation

We augmented the audio of each song in the datasets. The audio was first converted to a Mel spectrogram, which is a 2D array of time-frequency bins. The spectrogram was then augmented via a series of transformations adopted from (Park et al. 2019a), resulting in an augmented Mel spectrogram, which is an input to generative models.

We applied the following transformations in the presented order (see appendix G for details): the *frequency shift* shifts all frequency bins by a random amount; *frequency mask* fills some frequency bins with the mean value; *time mask* fills some time bins with mean value; *high frequency mask* also fills some frequency bins but such frequencies must be above a predetermined threshold; *frequency flip* reverses the order of frequency bins; *white noise* adds a Gaussian noise to all time and frequency bins; *time stretch* stretches all time bins.

The onset labels, which specify the existence of notes in the chart, were augmented by fluctuating the labels (Liang, Li, and Ikeda 2019). We also augment the beat information as explained in section 4.3.

4 GenéLive!

4.1 Audio Feature Extraction

Following Donahue, Lipton, and McAuley (2017), our model uses the Short-Time Fourier Transform (STFT) and Mel spectrogram of the audio. The STFT allows the model to capture features in the frequency domain. The window length and stride of STFT were both set to be 32 ms. The audio is sliced into chunks of 20 seconds.

The Mel spectrogram can capture perceptually relevant information in the audio data, and is a standard treatment in speech processing. It is also used in the DDC (Donahue, Lipton, and McAuley 2017). Following Hawthorne et al., Hawthorne et al. (2018), we set the number of the Mel bands to 229. We set the lowest frequency to 0 kHz and the highest to 16 kHz (0 and 3575 in the Mel scale). Accordingly, 229 evenly distributed triangular filters in the Mel scale are applied. We denote a Mel spectrogram by $S(t, f) \in \mathbb{R}$, where $t = 1, \ldots, T$ denotes the tth time bin, and $f = 1, \ldots, F$ denotes the fth frequency bin. We used $T = 20{,}000/32 = 625$ and $F = 229$ as mentioned above.

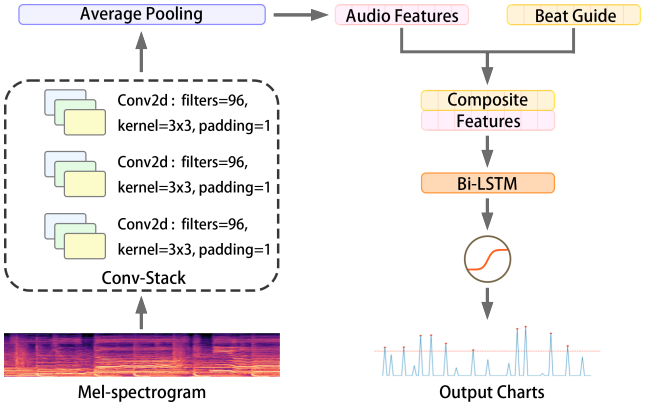


Figure 4: Overall architecture of our network.

4.2 Base Model

As shown in fig. 4, our base model follows the DDC (Donahue, Lipton, and McAuley 2017). The Mel spectrogram $S(t, f) \in \mathbb{R}$ is processed through the CNN layers to extract audio features $A(t, f) \in \mathbb{R}$. The audio features are concatenated with the game difficulty flag $D(t) =$ const. of 10 (Beginner), 20 (Intermediate), ..., 50 (Challenge) and the beat guide $G(t) \in \{0, 1, 2\}$. These two are fed to the BiLSTM layers (Graves and Schmidhuber 2005) to generate the chart $C(t) \in [0, 1]$.[5] Our improvements are explained in sections 4.3 and 4.4. Find more details of the model architecture and the corresponding parameters in appendix C.

Convolution Stack The main task for the convolution stack (or *conv-stack*) is to extract features from the Mel spectrogram using the CNN layers. The conv-stack comprises a standard CNN layer with batch normalization, a max-pooling layer, and a dropout layer. The activation function is ReLU. Finally, to regularize the output, we use an average-pooling layer.

4.3 Beat Guide

Although it had been rare to consider the positions of beats in the model, the beat is indeed crucial to the generation of the charts, as it is used by artists to evoke emotions. The beat guide is a trinary array whose length is the same as the number of time frames of the input audio. The first beat of each bar is indicated by 2, the other beats by 1, and non-beat frames by 0 (fig. 5). Each element indicates the existence of a beat at that frame. It is calculated from the BPM and time signature in the song metadata. The beat guide is fed as an input to the BiLSTM layers.

Note Timings Figure 6 shows how frequently each note timing appears in KLab's charts. The 4th note accounts for 70–90% of a chart, and the 8th takes up 10–20%; the 12th and 16th are marginal. This fact supports the effectiveness of the beat guide, as it provides hints for placing 4th notes.

[5]We employed BiLSTM based on our preliminary experiments. More recent architectures, including Transformer-XL (Dai et al. 2019), performed worse than BiLSTM for our task.

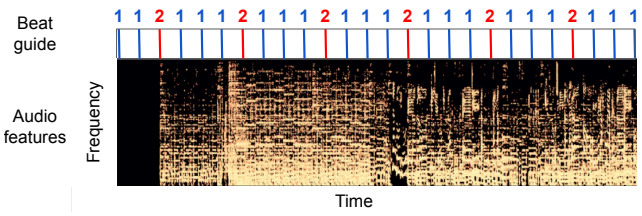


Figure 5: Beat guide in 4/4 time signature.

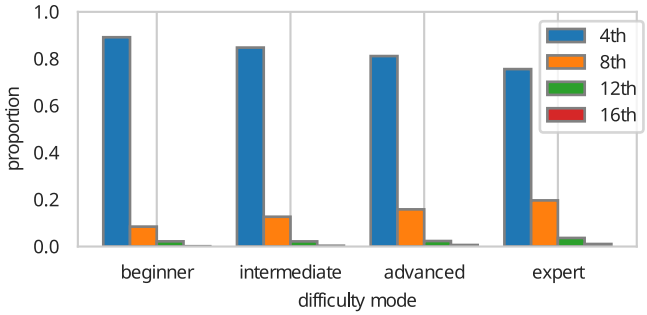


Figure 6: Note timings in "Love Live! All Stars."

It also hints that the multi-scale conv-stack's temporal max-pooling layers would be able to extract temporal dependencies of the 4th and 8th note scales.

Data Augmentation for Beat Guide Since the proposed model requires a beat guide as an extra input accompanied with a Mel spectrogram, it is also augmented. *Beat mask* drops beats in the section with given probability. The augmented guide is

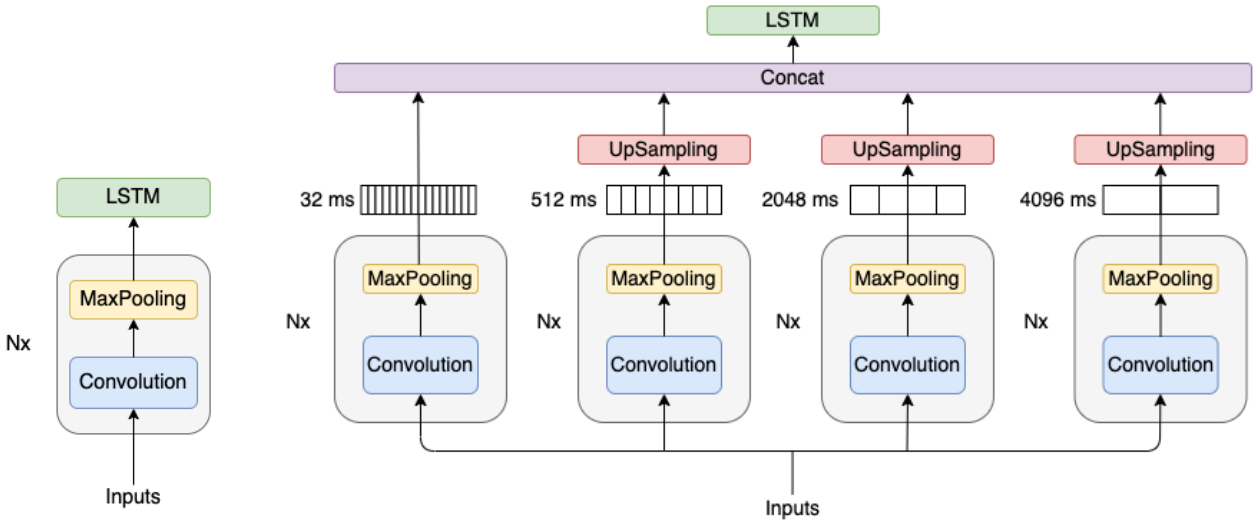
$$G(t) = \delta_t G_0(t), \tag{1}$$

where $G_0(t) \in \{0, 1, 2\}$ is the original beat guide at time step $t = 1, \ldots, T$, and $\delta_t \sim \mathcal{B}(1, p)$ is a random number drawn for each t from the binomial distribution with p being the probability of dropping a beat. The value of p was optimized to 0.123 by random search in the range $[0.1, 0.3]$. Finally, our model uses (S, G) as an input, where S is an augmented Mel spectrogram defined in eq. (9) in appendix.

4.4 Multi-Scale Conv-Stack

One key difference between the DDC and the present model is the structure of the conv-stack. In the model used in DDC, the convolution layers are applied repeatedly to the input of Mel spectrogram, whereas the max-pooling reduces the matrix size only along the frequency axis and not time (fig. 7a).

The present model uses four conv-stacks with different temporal resolutions. The stack with the highest resolution (stack 1) does not perform max-pooling along the temporal dimension. The process is the same as the conv-stack of the DDC. In stacks 2, 3, and 4, max-pooling is performed along the time dimension, and the length is reduced to $1/16$, $1/64$, and $1/128$, respectively. Finally, up-sampling is applied to stacks 2, 3, and 4, and the four matrices, which have the same length in the temporal dimension, are concatenated (fig. 7b). By doing so, we expect our model to extract



(a) DDC

(b) GenéLive!

Figure 7: Conv-stack architectures, previous vs. present.

not only short-term features (e.g., the attack of the percussion) but also long-term features (e.g., rhythm patterns and melodic phrases).

Note that unlike generic 2D multi-scale convolutions such as GoogLeNet (Szegedy et al. 2015), our temporal max-pooling does multi-scale pattern extraction explicitly and only along the time axis. Existing networks can be distracted by multi-scale patterns in frequency arisen by instruments such as piano or trumpet. More important, as agreed with our artists, is multi-scale temporal patterns.

The results of taking a combination of different conv-stacks are shown in fig. 8.

Effectiveness of Multi-Scale Analysis Figure 8 shows the effectiveness of multi-scale conv-stacks with different kernel sizes for max-pooling. The size 32 ms is the baseline chosen also by the DDC (without multi-scaling).

For candidates of max-pooling kernel size, we choose lengths with regular intervals in logarithmic scale to be musically meaningful length: 256 ms, 512 ms, 1024 ms, 2048 ms, and 4096 ms, each of which corresponds to the 8th note, 4th note, 2nd note, one bar, and two bars at 120 beats per minute (BPM) in 4/4 time signature.[6]

We can see that the 8th note (256 ms) typically worsens the learning compared to 32 ms. In this experiment combining 4 scales at maximum, the best one combines 32 ms, 512 ms (4th note), 2048 ms (one bar), and 4096 ms (two bars).

5 Experiments

For evaluation, we conducted a few experiments, while the business feedback is found in section 6. We took the ablation approach for evaluation, in which the *GenéLive!* model with all the presented methods applied was compared against models lacking some single component each.

As the characteristics of charts differ between different game titles, we conducted experiments separately for each game title. To spare pages, the text focuses on the results of the "Love Live! All Stars" dataset. With other datasets including another private dataset for "Utapri" in KLab and open datasets for Stepmania, the model exhibits similar results, confirming the genericity of the performance (See appendix B).

5.1 Training Methodology

We used a supercomputer to train the model with a vast range of hyperparameter configurations. In each training, we used the *BCE* as the loss function. Model parameters were updated using the *Adam* optimizer (Kingma and Ba 2014). The cosine annealing scheduler (Loshchilov and Hutter 2017) tuned the learning rate for better convergence. During the training, the dropout strategy was employed in both the fully connected layer and BiLSTM layer.

Tuning Hyperparameters The supercomputer let us conduct a grid search to determine the optimal combination of the following hyperparameters: the learning rate, $\eta_{\min}$ in the *cosine annealing scheduler* (Loshchilov and Hutter 2017), the choice of conv-stack, the width and scale of *fuzzy label* (Liang, Li, and Ikeda 2019), the dropout rate in the linear layer, the dropout rate in the RNN layer, the number of RNN layers, and the weighting factor in BCE loss.

To compare two sets of hyperparameters, we looked at the median of F_1-score$^{\mathrm{m}}$ (as explained in section 5.1) for the validation dataset.

To see hyperparameter values used in the experiment and candidates of grid-search, see table 14 in appendix E.

[6]The BPM 120 is the rough average in our data set (71 min / 230 max). The performance is insensitive enough to this factor (fig. 12 in appendix F).

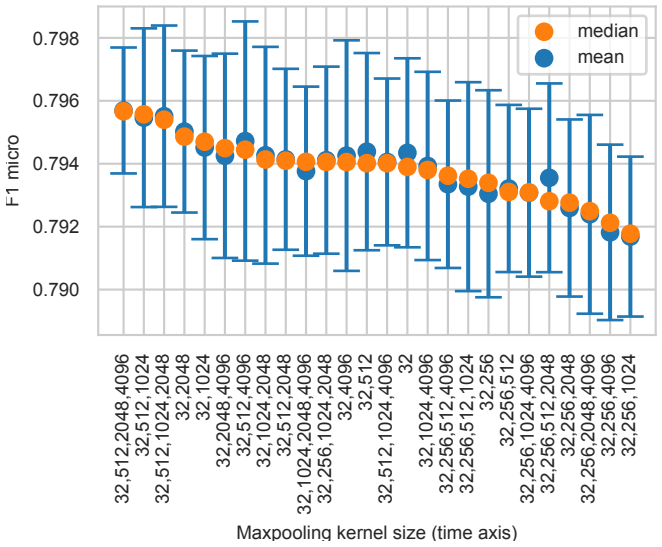


Figure 8: Experimenting with multi-scale temporal analysis. The results are sorted for the median of the F_1-score^m in descending order along the horizontal axis. The error bar shows 1σ of results. The baseline is a single max-pooling of kernel size 32 ms (placed in the middle of the horizontal axis) that was employed in the DDC model. There is a statistical significance in our multi-scale model (32, 512, 2048, 4096) improving the baseline (32) (Wilcoxon rank sum test, $W = 1703.5$, $p < 0.01$).

Computing environment We conducted the experiments using our supercomputer's NVIDIA Tesla P100 GPUs. We employed 64 of these GPUs to do the grid search in parallel. The implementation is based on `pytorch`, and to pre-process audio data `librosa` was employed. See `conda.yaml` file in the supplementary material for the complete list of software dependencies.

Data splitting The dataset was split into $8 : 1 : 1$ for training, evaluation, and testing sets with holdout employed. The dataset was first split into a few subgroups with similar BPM. The stratified K-folds cross-validator was then used to re-split each dataset into three.

Our model has no structural restriction on the time length of the input. In practice, however, a whole song cannot be supplied due to memory capacity constraints and the difference in the duration of songs. Thus, the audio, as well as the target, are cut into short chunks. Since the chunk length has room for exploration, we treated it as hyperparameters.

Metrics Following Donahue, Lipton, and McAuley (2017), we use the F_1-score as the evaluation metric.

Firstly, the F_1-score is calculated by averaging the results for each chart. We denoted the metric as F_1-score^c. Charts with different difficulty modes for the same song are considered distinct. Secondly, the score, F_1-score^m, is calculated by micro averaging. We considered the predicted note to be true positive if the note is placed within ± 50 ms around the ground truth. This is small enough for the shortest note (16th, which means 128 ms in case of 120 BPM) in our datasets.

We calculated the F_1-score for the test datasets using the median out of 10 experiments. For each of the 10, evaluation metrics were calculated for every step. the value with the highest performance for each evaluation metric was adopted as the performance of the model.

5.2 Results

Although a detailed discussion is to be found in section 5.3, we can see that our model consistently outperformed the state-of-the-art model (Donahue, Lipton, and McAuley 2017) as shown in table 1:

The F_1-score is explained in section 5.1. Although the metric is calculated for separate difficulties, the model was trained by being fed charts from all difficulties.

While we have already seen the results indicating the effectiveness of the multi-scale conv-stack in fig. 8, the same for the beat guide is shown in fig. 9.

Our decision to train a single model using charts from all difficulty modes is justified by the results in fig. 10, which is a comparison against training different models for different difficulty modes.

5.3 Discussion

Beat Guide Figure 9 shows that removing the beat guide significantly degraded the performance of the chart prediction task. Compared with the other two experiments, the beat guide is the most significant element in the present model.

The model's CNN seems to value locations where the sound is loud. If the beat guide is absent they tend to fail at other rhythm patterns like a repeating sequence with moderate volume.

Comparing the results for different difficulty modes, we can see that the effect of adding the beat guide to the input was larger for the easier difficulty modes. In general, charts of lower difficulty modes tended to consist of rhythms easier to capture. Keeping the beat of the song is one of the simplest rhythms, so in charts of lower difficulty modes, the note is often placed at the beat positions as shown in fig. 6. In turn, the easier difficulty mode of a chart, the more emphasis is placed on periodical rhythms, which CNN-based models are not good at, than the sound played in the music, and this may be the reason why the performance of the note generation task with easier difficulty modes is lower in previous studies such as (Donahue, Lipton, and McAuley 2017). Our experiments show that adding the beat guide to the input is a key to overcoming this weak point.

Multi-Scale Conv-Stack Figure 8 shows that the combination of multiple convolution stacks with appropriate max-pooling kernel size improves the performance, compared to the original conv-stack (denoted by 32 ms in the figure). This is the effect of the model being able to "look at" the time direction not only in the BiLSTM layer but also in the CNN layer by introducing time-axis max-pooling. The best combination of max-pooling kernel sizes is (32 ms, 512 ms, 2048 ms, 4096 ms). Specifically, compared to the original conv-stack, the stack 2 of our multi-scale conv-stack is able to take

Difficulty	F_1-score^m (mean $\pm$ SD) GenéLive!	DDC	p-value
Beginner	0.8664 ± 0.0036	0.7839 ± 0.0081	7.85×10^{-5}
Intermediate	0.7950 ± 0.0051	0.7457 ± 0.0039	7.85×10^{-5}
Advanced	0.7875 ± 0.0045	0.7491 ± 0.0044	7.85×10^{-5}
Expert	0.7955 ± 0.0038	0.7603 ± 0.0059	7.85×10^{-5}
all	0.8019 ± 0.0026	0.7514 ± 0.0046	7.85×10^{-5}

Table 1: Chart generation quality of the present model (GenéLive!) and the state-of-the-art (DDC) over 10 trials. Since in 10 trials the worst F_1-score of the proposed model is better than the best one of DDC in every difficulty mode, the one-sided Wilcoxon rank-sum test results in the possible smallest (identical) p-value, indicating that the proposed method statistically significantly outperforms over DDC.

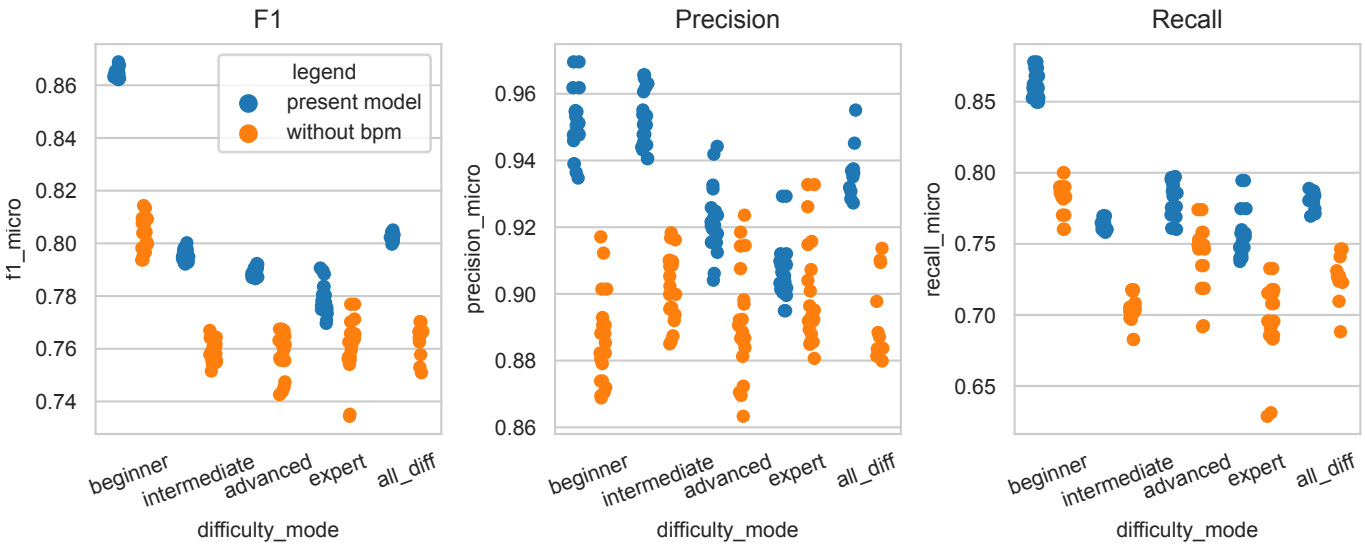


Figure 9: The beat guide enhanced the performance.

into account 16 times coarser information in the time direction (as discussed in section 4.4). Since the time resolution is 32 ms in our case (as explained in section 4.1), the second stack can consider 512 ms forward or backward, which corresponds to the length of a 4th note at 120 BPM. Similarly, our stack 3 sees 64 times coarser information, which amounts to one musical bar length at 120 BPM in 4/4 time signature, and our stack 4 sees 128 times coarser information, which amounts to two musical bar length.

Training With All Difficulties Before this work, it had been understood that the DDC was poor at generating charts for easier difficulties (Donahue, Lipton, and McAuley 2017), yet it had been unclear what kind of improvements can be effective. That is to ask, is it better to let a *single* model instance consume charts having multiple difficulties or, instead, *multiple* instances consume them, where each instance is specialized for a certain difficulty mode? On one hand, the similarity between charts from different difficulties for the same song could work as a hint. On the other hand, however, the network might be confused by the same song resulting in different charts. Another question regarding the poor performance of the DDC on easier difficulties is whether the fundamentals of its design had a flaw in the first place.

Figure 10 shows that when all difficulty modes were trained with a single model, the performance improved. This result implies the effectiveness of the CNN for learning charts of varying difficulties. This work is the first to reveal that learning different difficulties with a single model instance in fact outperforms the other. The same experiments of ours also indicate that the DDC's approach, which we adapted, indeed is capable of predicting easy difficulties, although our model had received improvements.

What is the reason behind, though? In the next paragraph, we show our analysis that indicates a strong inclusion relation between charts for different difficulties but of the same song, which we suspect is the reason.

Inclusion Relation Between Difficulties We view a chart as a bit string representing the existence of a tap for frames in a quantized time. We define the *inclusion rate* as

$$I(s,t) = \frac{\|s \otimes t\|}{\|s\|} \tag{2}$$

for strings s and t, where $\otimes$ is the bit-wise AND product, and $\|\cdot\|$ is the number of 1s in a string. Inclusion rates (averaged

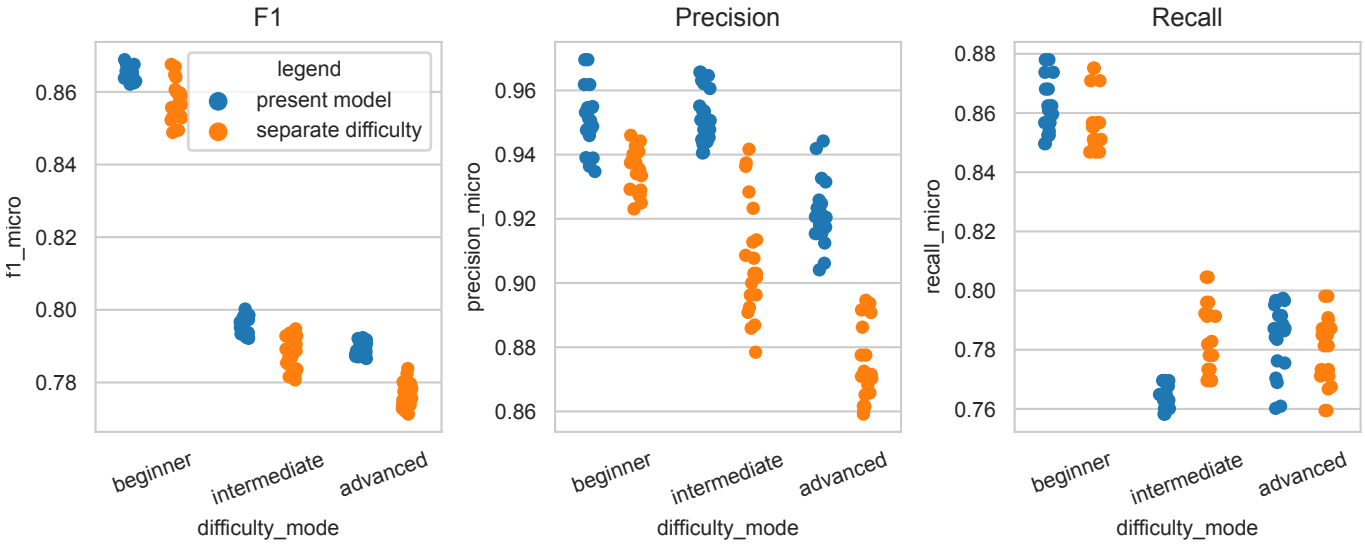


Figure 10: Performance with training using datasets from all difficulties vs a single difficulty.

t \ s	Beginner	Intermediate	Advanced	Expert
Beginner	100%	64.7%	49.8%	39.0%
Intermed.	97.4%	100%	76.2%	59.4%
Advanced	97.4 %	98.8 %	100%	73.5%
Expert	98.6 %	98.2%	98.6%	100%

Table 2: Inclusion rates in "Love Live! All Starts."

over songs) in table 2 show that different difficulty modes actually share a large portion of notes. Especially, more than 97% of notes in an easier chart appears also in a harder one, indicating that training with easier charts contributes to the prediction of harder ones. This pattern is also found in all datasets we targeted (see appendix A), and thus seems universal.

6 Business Feedback

Since the first deployment of this chart generation system (July 2020), the artist team has used the system to create charts for all 110 songs released in "Love Live! All Stars" (82 songs had been released before the period). The present collaboration cut down the chart creation time for the artist team by half. About 20 hours of work are saved per song, whereas it used to take about 40 hours per song. For the Beginner and Intermediate difficulty modes, the charts generated by our model can be used with minor modifications. The artist team uses our model also for the more difficult game modes.

Artists employed the timing for notes generated by the model mostly without alteration. Figure 11 compares the first 8 bars of an automatically generated chart with the released version which was manually modified from the generated one. Of the 22 auto-generated notes, 21 were accepted as they were, and only one was not used, while three notes were added. Such a high-quality chart is mainly brought by

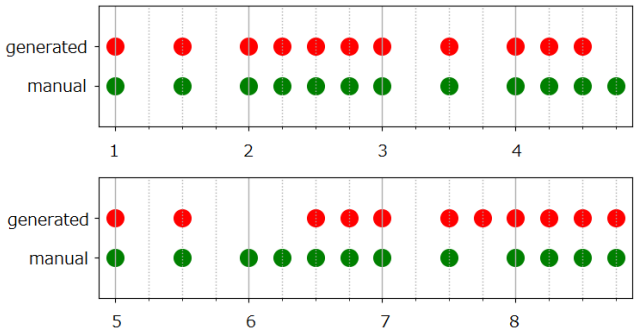


Figure 11: A generated chart for the Advanced difficulty mode and its manually-modified version (the first 8 bars).

our model's novel ability to recognize musical structures. As you can listen to the song on YouTube[7], this 8-bar intro repeats a 2-bar phrase where the phrase is altered every repetition. Our model reflected such an altered repetition of a 2-bar pattern in the generated chart (fig. 11). This is true for the whole of the song, see fig. 13 in appendix H.

7 Related Work

Early chart generation tended to use a rule-based algorithm (O'Keeffe 2003) or genetic algorithm (Nogaj 2005). The DDC by Donahue, Lipton, and McAuley (2017) improved the quality by employing the deep neural network. Lin, Xiao, and Riedl (2019) used a multilayer feed-forward network to generate charts, synchronizing the notes with instrumental sounds. The model by Tsujino and Yamanishi (2018) alters the difficulty of an input chart – this work instead generates

[7]The song is available on https://youtu.be/MpAUJ36fq3g, and its portion from intro to first chorus (time range 0:16–2:07) is played in our rhythm game.

the first draft chart (without an input chart).

Onset detection based on neural networks is studied for speech recognition, where BiLSTMs (Eyben et al. 2010), CNN approach (Schlüter and Böck 2014), and multi-resolution feature representation are established. Schlüter and Böck (2014) demonstrated a CNN-based approach. Likewise, music information retrieval has also seen benefits of neural networks (Humphrey and Bello 2012; Boulanger-Lewandowski, Bengio, and Vincent 2013; Ullrich, Schlüter, and Grill 2014).

In addition, various deep generative models are used to create game content and assets. *Procedural content generation via machine learning*, a model for generating game content using trained machine learning models, can generate a variety of game content such as items, maps, and rules (Summerville et al. 2018; Guzdial et al. 2018). Hastings, Guha, and Stanley (2009) proposed an automatic content generation that learns players' preferences based on their past play history and generates new graphical and game content during gameplay. In addition, Green et al. (2021) proposed a framework generating tutorial text by discovering whether it is effective to win or lose a game. Tilson and Gelowitz (2019) investigated generating image assets for games using unsupervised learning such as GAN and VAE.

Volz et al. (2018) have generated various levels of Super Mario Bros. using GAN. Park et al. (2019b) used GAN to generate levels for a computer science educational game.

There is a method to synthesize high-quality, realistic full-body animations of 3D characters using deep learning (Ding 2021). It is possible to express emotions and generate facial expressions and movements corresponding to the NPC's personality and profession. A model has been developed to create a face mesh from a photograph of a face and generate a face model of a 3D character (Li 2021).

In addition, there are examples of successful use of deep learning to reduce man-hours in the game development process, such as the development of AI for mini-games in MMORPGs by applying AlphaZero (Lei 2021) and the improvement of game frame rates by applying deep super-resolution in the time direction (Edelsten 2021).

8 Conclusions

We assisted in chart creation at KLab Inc. by establishing a new deep generative model, while the model ended up being in fact versatile for more generic datasets. The model successfully generated charts for easier difficulty modes, filling the quality gap between easier and harder game modes, which was a challenge that had been admitted by the authors of the state-of-the-art model DDC. This was achieved by utilizing two techniques, (i) the beat guide and (ii) the multi-scale conv-stack. KLab successfully cut the business cost by half.

Acknowledgements

The authors would like to say *arigatou* to every member of μ's, Aqours, Nijigaku, Liella!, Hasunosora and all school idols, as well as Lovelivers (fans) over the world, whose energy of *daisuki* always encouraged us and drove our research. We are especially grateful to Lingjian Wang and Keita Yamamoto for their development and operation of our chart generation system, Chiharu Shineha, Yuko Okada and Takahiko Anbo for their insightful comments from the professional viewpoint of chart creation, and Chris Donahue for kind help on software licensing. This work was supported by JSPS KAKENHI Grant Number 20K19809 and FY2021 IMI Joint Usage Research Program General Workshop (II) "Fiber Topology Meets Applications 2" (20210013). The computation was carried out using the computer resource offered under the category of Intensively Promoted Projects by Research Institute for Information Technology, Kyushu University.

References

Boulanger-Lewandowski, N.; Bengio, Y.; and Vincent, P. 2013. Audio Chord Recognition with Recurrent Neural Networks. In *Proceedings of the 14th International Society for Music Information Retrieval Conference*, 335–340. Curitiba, Brazil: ISMIR.

Dai, Z.; Yang, Z.; Yang, Y.; Carbonell, J. G.; Le, Q. V.; and Salakhutdinov, R. 2019. Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. In Korhonen, A.; Traum, D. R.; and Màrquez, L., eds., *ACL (1)*, 2978–2988. Association for Computational Linguistics. ISBN 978-1-950737-48-2.

Ding, Y. 2021. Machine Learning Summit: Full-Body Animation Generation for Expressive NPCs. Accessed on February 10, 2022.

Donahue, C.; Lipton, Z. C.; and McAuley, J. 2017. Dance Dance Convolution. In Precup, D.; and Teh, Y. W., eds., *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, 1039–1048. Sydney, Australia: PMLR.

Edelsten, A. 2021. Machine Learning Summit: 3D Parametric Face Model and Its Applications in Games. Accessed on February 10, 2022.

Eyben, F.; Böck, S.; Schuller, B. W.; and Graves, A. 2010. Universal Onset Detection with Bidirectional Long Short-Term Memory Neural Networks. In Downie, J. S.; and Veltkamp, R. C., eds., *Proceedings of the 11th International Society for Music Information Retrieval Conference, ISMIR 2010, Utrecht, Netherlands, August 9-13, 2010*, 589–594. Utrecht, Netherlands: International Society for Music Information Retrieval.

Graves, A.; and Schmidhuber, J. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural networks*, 18(5-6): 602–610.

Green, M.; Khalifa, A.; Barros, G.; and Togellius, J. 2021. "Press Space to Fire": Automatic Video Game Tutorial Generation. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 13(2): 75–80.

Guzdial, M.; Reno, J.; Chen, J.; Smith, G.; and Riedl, M. 2018. Explainable PCGML via Game Design Patterns. In Zhu, J., ed., *Joint Proceedings of the AIIDE 2018 Workshops*

co-located with 14th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2018), Edmonton, Canada, November 13-14, 2018, volume 2282 of *CEUR Workshop Proceedings*. Edmonton, Canada: CEUR-WS.org.

Hastings, E. J.; Guha, R. K.; and Stanley, K. O. 2009. Automatic Content Generation in the ¡emphasis emphasistype="italic"¿Galactic Arms Race¡/emphasis¿ Video Game. *IEEE Transactions on Computational Intelligence and AI in Games*, 1(4): 245–263.

Hawthorne, C.; Elsen, E.; Song, J.; Roberts, A.; Simon, I.; Raffel, C.; Engel, J. H.; Oore, S.; and Eck, D. 2018. Onsets and Frames: Dual-Objective Piano Transcription. In Gómez, E.; Hu, X.; Humphrey, E.; and Benetos, E., eds., *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France, September 23-27, 2018*, 50–57. Paris, France: International Society for Music Information Retrieval.

Humphrey, E. J.; and Bello, J. P. 2012. Rethinking Automatic Chord Recognition with Convolutional Neural Networks. In *11th International Conference on Machine Learning and Applications, ICMLA, Boca Raton, FL, USA, December 12-15, 2012. Volume 2*, 357–362. Florida, USA: IEEE.

Kingma, D. P.; and Ba, J. 2014. Adam: A Method for Stochastic Optimization. Cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

Lei, Z. 2021. Machine Learning Summit: Applying 'AlphaZero' to Develop AI in Turn-Based Card Games. Accessed on February 10, 2022.

Li, P. 2021. Machine Learning Summit: 3D Parametric Face Model and Its Applications in Games. Accessed on February 10, 2022.

Liang, Y.; Li, W.; and Ikeda, K. 2019. Procedural Content Generation of Rhythm Games Using Deep Learning Methods. In van der Spek, E.; Göbel, S.; Do, E. Y.-L.; Clua, E.; and Baalsrud Hauge, J., eds., *Entertainment Computing and Serious Games*, 134–145. Cham: Springer International Publishing. ISBN 978-3-030-34644-7.

Lin, Z.; Xiao, K.; and Riedl, M. 2019. GenerationMania: Learning to Semantically Choreograph. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 15, 52–58.

Loshchilov, I.; and Hutter, F. 2017. SGDR: Stochastic Gradient Descent with Warm Restarts. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. Palais des Congrès Neptune, Toulon, France.

Nogaj, A. F. 2005. A genetic algorithm for determining optimal step patterns in Dance Dance Revolution. Technical report, Technical report, State University of New York at Fredonia.

O'Keeffe, K. 2003. Dancing Monkeys (Automated creation of step files for Dance Dance Revolution). Technical report, Imperial College London, London, United Kingdom.

Park, D. S.; Chan, W.; Zhang, Y.; Chiu, C.-C.; Zoph, B.; Cubuk, E. D.; and Le, Q. V. 2019a. SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition. In *Interspeech 2019*. ISCA.

Park, K.; Mott, B. W.; Min, W.; Boyer, K. E.; Wiebe, E. N.; and Lester, J. C. 2019b. Generating Educational Game Levels with Multistep Deep Convolutional Generative Adversarial Networks. In *2019 IEEE Conference on Games (CoG)*, 1–8. London, UK: IEEE.

Schlüter, J.; and Böck, S. 2014. Improved musical onset detection with Convolutional Neural Networks. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 6979–6983. Florence, Italy: IEEE.

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.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1–9.

Tilson, A.; and Gelowitz, C. M. 2019. Towards Generating Image Assets Through Deep Learning for Game Development. In *2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE)*, 1–4. Edmonton, AB, Canada: IEEE.

Tsujino, Y.; and Yamanishi, R. 2018. Dance Dance Gradation: A Generation of Fine-Tuned Dance Charts. In Clua, E.; Roque, L.; Lugmayr, A.; and Tuomi, P., eds., *Entertainment Computing - ICEC 2018 - 17th IFIP TC 14 International Conference, Held at the 24th IFIP World Computer Congress, WCC 2018, Poznan, Poland, September 17-20, 2018, Proceedings*, volume 11112 of *Lecture Notes in Computer Science*, 175–187. Poznan, Poland: Springer.

Ullrich, K.; Schlüter, J.; and Grill, T. 2014. Boundary Detection in Music Structure Analysis using Convolutional Neural Networks. In Wang, H.; Yang, Y.; and Lee, J. H., eds., *Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR 2014, Taipei, Taiwan, October 27-31, 2014*, 417–422. Taipei, Taiwan: International Society for Music Information Retrieval.

Volz, V.; Schrum, J.; Liu, J.; Lucas, S. M.; Smith, A. M.; and Risi, S. 2018. Evolving mario levels in the latent space of a deep convolutional generative adversarial network. In Aguirre, H. E.; and Takadama, K., eds., *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2018, Kyoto, Japan, July 15-19, 2018*, 221–228. Kyoto, Japan: ACM.