TaikoNation: Patterning-focused Chart Generation for Rhythm Action Games

Emily Halina
University of Alberta
Edmonton, Canada
ehalina@ualberta.ca

Matthew Guzdial
University of Alberta
Edmonton, Canada
guzdial@ualberta.ca

ABSTRACT

Generating rhythm game charts from songs via machine learning has been a problem of increasing interest in recent years. However, all existing systems struggle to replicate human-like patterning: the placement of game objects in relation to each other to form congruent patterns based on events in the song. Patterning is a key identifier of high quality rhythm game content, seen as a necessary component in human rankings. We establish a new approach for chart generation that produces charts with more congruent, human-like patterning than seen in prior work.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; • **Applied computing** → **Sound and music computing**.

KEYWORDS

rhythm games, neural networks, procedural content generation

ACM Reference Format:
Emily Halina and Matthew Guzdial. 2021. TaikoNation: Patterning-focused Chart Generation for Rhythm Action Games. In *The 16th International Conference on the Foundations of Digital Games (FDG) 2021 (FDG'21), August 3–6, 2021, Montreal, QC, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3472538.3472589

1 INTRODUCTION

Procedural Content Generation via Machine Learning (PCGML) is defined as the generation of novel game content through machine learning models that have been trained on previously existing game content [37]. Previously, many PCGML approaches have been applied to 2D side-scrolling platformer games, where they were used to generate new levels based on existing training data. However, many underexplored game genres exist that may benefit from PCGML approaches. In this paper we focus on one such game genre: rhythm games. Rhythm games challenge the player to hit a series of predetermined inputs in time to a given song within a level. These levels, known as charts, are traditionally handcrafted by individual authors, requiring both expertise and several hours of work to complete. Games such as Rock Band 4 have over 1700 charts available as downloadable content [19], meaning this quickly becomes a large-scale problem. Chart creation requires specialized design knowledge that can take several months to hone and develop [1, 16]. This makes chart creation inaccessible to many people who wish to see their favourite songs charted. We focus on the problem of procedural chart generation via machine learning, generating a novel chart based on a given input song with a machine learned model. A trained model for chart generation could help to make chart creation more accessible to a wider audience, and speed up the chart creation workflow for developers.

Chart generation has been attempted in prior work, but there are open problems that remain to be solved. For example, there is the problem of onset detection, which refers to analyzing a piece of audio to locate the beginning of each musical note or beat within it. Onset detection has been the central focus of prior work involving chart generation, notably Dance Dance Convolution [6]. However, just placing notes at appropriate times is not sufficient to create a compelling, engaging chart, especially at a high difficulty level [16]. For example, high level charts include "patterning," the placement of game objects in relation to each other to form congruent patterns based on events in the song [1]. Patterning is a fundamental component of highly-rated charts that allows individual authors to express their unique interpretations of a song [1]. The problem of training a model to recognize the appropriate times to place notes to create consistent, human-like patterns is still largely unexplored.

In this paper, we focus on the task of chart generation for the game Taiko no Tatsujin (abbreviated as Taiko) [40]. Taiko is a long running rhythm game franchise modelled on simulating the playing of a taiko drum. We chose Taiko due to its heavy emphasis on percussive rhythm, which made it a natural fit to approaching the problem of generating charts containing human-like patterning. After curating a dataset of 100 highly rated charts, we trained a Long Short Term Memory Recurrent Neural Network (LSTM RNN) to translate music to Taiko charts. Unlike prior work [6, 21, 22], we predict multiple outputs simultaneously, biasing the model towards longer-form patterns. We call our system "TaikoNation," a portmanteau of the name of a popular osu!Taiko mascot and "generation." Our system creates more congruent, human-like patterning than seen in prior chart generation work.

In this paper, we present the following contributions:

- An LSTM architecture for generating novel Taiko charts based on arbitrary audio.
- A curated dataset of 110 Taiko charts along with their corresponding song data in a novel format suitable for machine learning. We also include 10 Dance Dance Revolution charts in this format used for comparison.[1]

[1] https://github.com/emily-halina/TaikoNationV1

FDG'21, August 3–6, 2021, Montreal, QC, Canada

ACM ISBN 978-1-4503-8422-3/21/08…$15.00
https://doi.org/10.1145/3472538.3472589

- Experiments comparing our model to an existing approach: Dance Dance Convolution [6].

We begin by examining prior works involving chart generation and other relevant topics. Following this, we cover a full system overview detailing the architecture used for the model and the steps taken to process the input and output. We define evaluation methods for chart generation models centered around both onset detection and different notions of patterning, and compare our model's results against Dance Dance Convolution [6]. Finally, we discuss the future potential uses of the model beyond this paper in a co-creative context.

2 RELATED WORKS

2.1 PCGML Approaches for Sequence Generation in Games

PCGML, the generation of new content for a game through machine learning models trained on existing game content, has been applied to sequence generation problems like ours in the past. Many prior works have made use of sequence-to-sequence models such as LSTMs, which are commonly used for sequence generation problems [15, 37, 38]. LSTMs have been used to generate many different types of content outside of rhythm game charts like Super Mario Bros. levels and Magic the Gathering cards [35, 36]. We similarly make use of an LSTM trained for our task of Taiko chart generation.

A common approach across PCGML is to generate subsections of the final desired content in order to better leverage existing data [27, 32]. This approach has been used in multiple game domains to generate subsections of levels which are then strung together [29, 39]. We similarly make use of this approach, but focus on generating longer subsections than prior work in PCGML chart generation [6, 21, 22].

2.2 Chart Generation & Music-Related PCG without Machine Learning

Early attempts at chart generation for rhythm games utilized rule-based techniques [26] and genetic algorithms [24] to determine where to place game objects within charts. There is also prior work utilizing PCG techniques with a rhythmic focus for other game domains, such as platformers [30, 31]. Prior work also includes PCG based on user input, altering or blending songs together based on player actions [11, 17]. There are also several commercial games focused around the use of PCG techniques on arbitrary music to generate content [8, 9].

2.3 Chart Generation with Machine Learning

There have been a number of instances of prior work on the topic of PCGML chart generation for rhythm games. These approaches break the chart generation pipeline into multiple distinct segments and handle the problem in pieces. Donahue's Dance Dance Convolution [6] utilized two separate machine learning models to handle the tasks of note placement (deciding when to place a note in the song) and step selection (deciding which input to assign to each note) respectively. By note we indicate a given musical onset that is assigned a game object that requires player input. Liang's work Procedural Content Generation of Rhythm Games (PCGoRG) [21], further extended the model used in Dance Dance Convolution, improving the onset detection model through the usage of larger stride windows (called "fuzzy labelling"). Lin's GenerationMania [22] focused on sample classification and selection. This additional focus is due to the nature of the game domain for Generationmania: Beatmania IIDX, which is a "keysounded" game. In keysounded games, the player's input directly influences the song, as each note directly represents a musical sample within the track. This causes the player to generate the song live as they play the chart, which adds a new dimension to the problem of chart generation for IIDX. In contrast, Taiko is a non-keysounded game, so this problem is not present within our work.

There are multiple standout differences between our approach and these three prior attempts at chart generation. Notably, in these examples of prior work, the problem of chart generation was divided into multiple parts, with onset detection and game object selection being split into separate problems. In our model, the two problems are handled simultaneously with the goal of creating a stronger link between the timing and object type of each given note. In addition to this, each of the three prior approaches have a built in notion of difficulty, attempting to adjust the density of a given output chart based on the desired difficulty. In contrast, our work focuses on the expert level of difficulty, because of our focus on the more complex relationships and patterns present between notes at a high difficulty. In general, PCGML approaches have proven less effective for lower difficulties so far, as valid low level charts have substantially greater sparsity. While these examples of prior work have a notion of prior context due to implementations of systems like "chart summaries," which contain selected information from the previous measures of a chart, they still predict output for each timestep individually. Our model has a sense of structure and summary embedded into it, as we predict over multiple timesteps worth of input at a time. When combined, we hypothesize that these differences will lead to a better handling of patterning than these prior approaches, leading to more congruent placement of notes in relation to each other.

3 SYSTEM OVERVIEW

Figure 1 depicts the chart generation pipeline from the initial dataset to the creation of a Taiko chart based on arbitrary song input. After giving a broad overview of the pipeline, we will go into detail in the following subsections. We begin with a curated dataset of 100 Taiko charts, with song data in the form of a 192kbps mp3 audio file and note data contained in a .osu file [4] for each chart. We slice the audio file into 23 millisecond (ms) segments and extract the audio features from each segment. We parse and convert the note data from the .osu file into a new representation for training, which we describe below. This pre-processed data is fed into our LSTM DNN architecture in bundles of 16 segments at a time, and the model predicts the next 4 notes of the chart based on the previous note and chart data. After training on the dataset, we use this model to generate predictions on a sliding window of 16 x 23 ms across a given input song, averaging the predictions for each timestamp together. These averaged predictions are converted back into a playable format with some light post-processing, creating a chart.

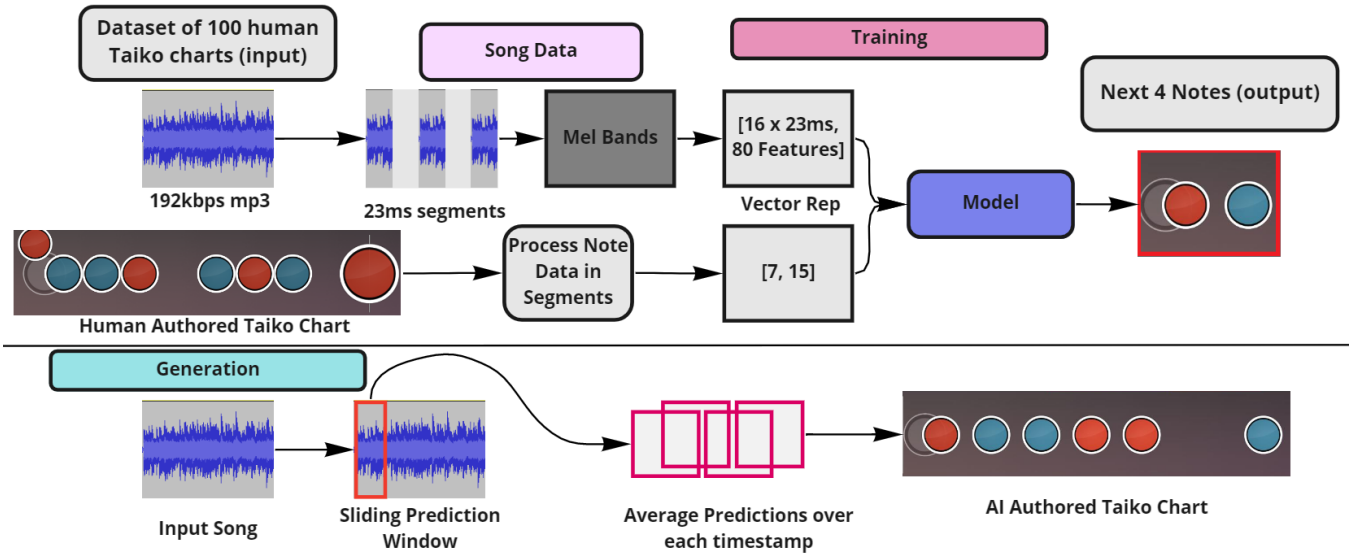

Figure 1: System Overview of our chart generation pipeline. We use a LSTM DNN architecture trained on a curated, human-authored dataset to predict the next four notes of a chart from a combination of song and note data. We use this model to make predictions on a given input song over a sliding window, averaging the predictions for each timestamp to output a chart.

3.1 Human-Like Patterning

Our system attempts to replicate the notion of "human-like patterning," or shortly "patterning." Patterning is not a well-defined term, and we are not attempting to define it here. Instead, we give our interpretation of patterning and provide some examples of what patterning entails in the context of this paper.

The main aspect of patterning considered in this paper is the placement of game objects (notes) in relation to each other to form congruent patterns based on events in the song [1]. As an example, three notes placed directly next to each other on the timeline form a triplet, which is a specific rhythmic pattern. Patterns can occur on multiple scales, from over a particular measure to over an entire chart. For example, a specific recurring set of notes over the course of a song could be considered a higher-level pattern, analogous to a motif in music. The congruence between notes within patterns can be based on rhythmic placement as well as the types of notes used. In Taiko, these two congruences are highly interconnected, with the majority of patterns found in high quality human-authored charts utilizing both note type and rhythmic structure simultaneously. For instance, chart authors may use the same rhythmic structures multiple times in a row with different note types. This can lead to patterns that feel completely different to play despite having the same rhythmic structure. We focus mainly on replicating the rhythmic half of this problem while also attempting to match the appropriate distribution of each type of note object found in human-authored charts.

3.2 Data

Our goal is to investigate patterning, and thus we need a collection of high quality charts that showcase intricate, song-appropriate patterning. The dataset was collected from a community run database of approved charts for Taiko [14]. These charts all follow a set Ranking Criteria [5], requiring them to be scrutinized by experienced creators before they are approved. Since these reviewers value patterning [5], these charts are more likely to contain the high quality patterning we are attempting to model. We further verified this was the case based on an author's expert knowledge of patterning. We sorted these approved charts by user rating from high to low, and took any chart above a certain difficulty threshold. This threshold was set at a relatively high level of difficulty to ensure the dataset contained a varied collection of dense, interesting patterns. If there was more than one chart for the same song that was above this difficulty threshold, the higher rated of the two was chosen. For this initial exploration, we collected 100 charts with a 90-10 split between the training and validation set. This resulted in approximately 1.3 million elements to train on once processed. These 100 charts represent a large number of the database's top rated charts, and cover a wide span of genres and charting styles.

Our approach to audio representation is similar to Dance Dance Convolution's [6, 28], with a few differences based on game domain specifications and our focus on patterning. Each chart's corresponding audio was initially a 192kbps mp3 file. The specific audio quality is due to Ranking Criteria specification [5]. While there is a concern for audio artifacts due to this low bitrate, each audio file in the dataset has been reviewed by multiple chart authors. This ensures the best possible audio quality as a part of the aforementioned ranking process. We cut the audio into 23ms segments, converting each segment to a monaural, or single channel, .wav file for processing. 23ms segment sizes were chosen after an analysis of note placements relative to beats per minute (BPM) within the dataset. Specifically, 23ms is roughly the distance between two 1/64th notes

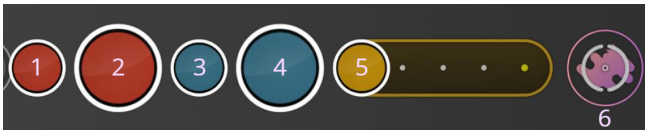

Figure 2: Depiction of each note type present within Taiko. In order these are: small Don, big Don, small Kat, big Kat, Drumroll, and Denden.

at 163 BPM, and offers a reasonable divide for most BPMs outside of extreme ranges.

We perform a short-time Fourier transform (STFT) on each of our extracted segments, using a window and stride length of 23ms to match our segment length. STFTs are a fundamental tool of signal analysis, and have been used in onset detection frameworks similar to ours in prior work [2, 28]. While Dance Dance Convolution used a multiple-timescale transform for its onset detection pipeline, we instead used only one timescale length. This is because we use a larger stride length, which along with appending the previous note data provides the rhythmic context that would otherwise be given by using multiple longer timescales [13]. We then compress the dimensionality of our spectra down to 80 frequency bands by applying a Mel-scale filterbank [34]. This allows us to extract the key features from the STFT results into evenly spaced divisions, which are then scaled logarithmically to account for human perception of loudness [6]. We used the ESSENTIA audio processing library [3] for this task, normalizing each frequency band to zero mean and unit variance. After this processing, we append the previous 15 segments of audio data to each segment, giving a dimensionality of (16 x 80) per segment, representing 368ms of song data.

The chart information from the dataset comes in the form of .osu files, a human-readable file format containing information about the game objects used in the chart and their timestamps [4].

We parsed this file format, extracting the note data in 23ms timesteps and encoding the various game objects into a 7-length one-hot representation. This representation included an entry for "no note," along with all 6 of the main object types found in Taiko. These are:

- Small / Big Don: 1 & 2 in Figure 2
  One of two main types of notes. Dons have two distinct keys (or the center of the drum) associated with them. Small Dons require you to hit one key, and big Dons require you to hit both keys.
- Small / Big Kat: 3 & 4 in Figure 2
  The second of two main types notes. Kats have two distinct keys (or the rim of the drum) associated with them. Small Kats require you to hit one key, and big Kats require you to hit both keys.
- Drumrolls: 5 in Figure 2
  A rare object type that requires drumming in time to a consistent 1/4 rhythm for a given duration. Any keys or area of the drum can be used during these segments.
- Dendens: 6 in Figure 2
  A rare object type that requires the player to hit the keys / drum a set number of times before time is up. These are not associated with the rhythm of the song, and can be considered analogous to "drum fills".

At the timestamps where the model should predict a note, we represent that note by filling the index of the given note vector with 1s. This led to a dimensionality of (7 note features x 15 23ms timestamps) for each segment of note data.

3.3 Architecture

A visualization of our LSTM DNN architecture can be seen in Figure 3. The input consists of 16 segments of song data representing 368 ms of audio and 15 segments of the corresponding previous note data. We run the song data through a convolutional layer with 16 filters using a rectified linear unit (ReLU) activation function. Convolutional layers are commonly used to parse sound data for many signal processing tasks including onset detection [20, 28].

ReLU has been demonstrated to better model complex, non-linear relationships than other activation functions [41]. For this reason we use ReLU activation for the remaining layers, excluding the final output layer. This is followed by a 80% dropout layer, then a max-pooling layer. Dropout layers are helpful to reduce overfitting, allowing us to learn more general models [33]. The 80% value arose due to a class imbalance, as roughly 83% of our notes are the "no note" class. Max-pooling layers were used as we wanted to encourage the network to focus on large onsets for note placement. This is followed by another convolution layer with 32 filters, followed by another max-pooling layer.

We pass this result through a fully-connected layer with 128 nodes, which is reshaped to an (8 x 16) vector. This is done to combine the CNN output with the note data by performing by-element multiplication on each segment. We chose a by-element multiplication instead of treating these as two separate channels in order to tightly associate the tasks of onset detection and object selection, which were treated separately in prior chart generation work [6, 21, 22]. Since the final segment does not have note data as part of our input, we multiply that segment by a vector of 1s as a placeholder.

From here, we run our combined data through two LSTM layers with 64 nodes, 80% dropout, and ReLU activation. LSTM layers were chosen due to their suitability for sequence-to-sequence learning tasks, and have been demonstrated to perform well on timing dependant audio tasks [10, 38]. Finally, we pass this data through a softmax activated fully connected output layer with 28 nodes. Softmax activation is typically used when predicting one particular output class, as it acts as a probability distribution across each of the possible output classes [23]. This is reshaped to a (4 x 7) vector, representing predictions for which gameplay objects to place at the next 4 given timestamps.

We trained the model using the Adam optimizer and categorical cross-entropy as a loss function. Taiko charts have a large amount of variance, as there can be several valid charts for the same song, which is why we chose Adam over SGD [18]. Initially, we trained the model for 10 epochs with a learning rate of 0.00001 and a batch size of 16. These values were experimentally verified on our validation set. After this initial training, we dropped the batch size down to 1, turning down the learning rate to 0.000005 and retraining one epoch at a time until weight explosion, taking the prior epoch's

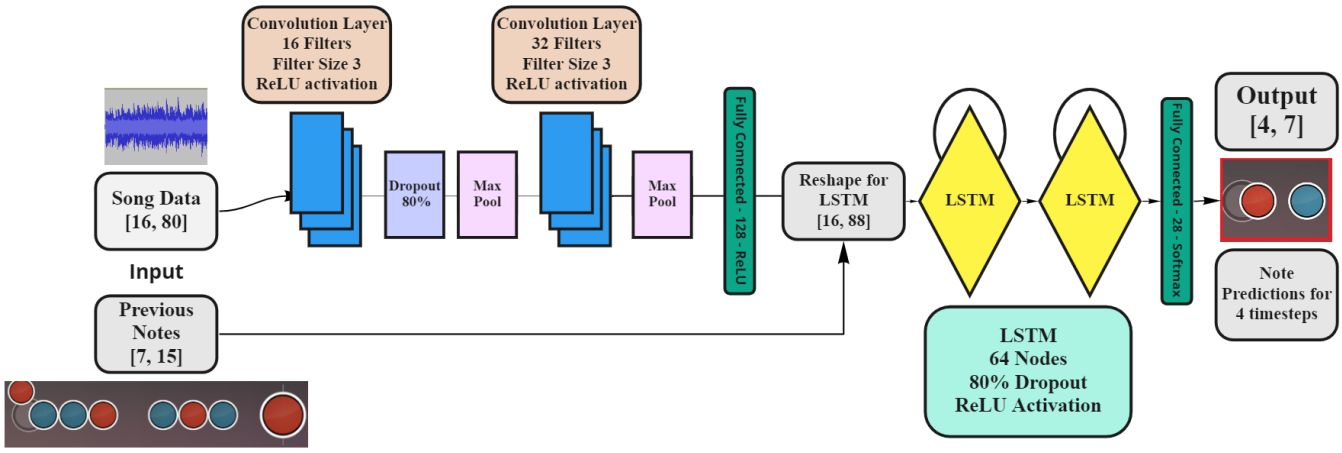

Figure 3: A visualization of our LSTM DNN Architecture. Two convolutional layers are used to extract features from the input segment of song data, which represents 368ms of the input song. These features are then combined with the previous note data and passed through two LSTM layers before outputting note predictions for the following four timestamps in the song.

model. This process took 4 additional epochs. While this is not the most robust training methodology, we employ it for this initial exploration of the approach due to the high variance in the data.

In total, the model trained for approximately 2 and a half hours on a single machine using an AMD Ryzen 5 3600x CPU and NVIDIA GeForce 1080 Ti GPU.

3.4 Generation

Our chart generation pipeline begins with a full song as input. We begin by processing the song, slicing it into 23ms segments and converting it to our uniform monaural .wav file format. Next, each slice of the song has its features extracted as per the processing described in Section 3.2. After the song data has been prepared and processed, we use the model to make predictions on the song data using a sliding window of 16 segments, or 368ms. We initially use placeholder note data for the first 16 segments, which takes the form of a vector of 0s. This is because there are very rarely notes in the immediate starting moments of the audio file, so we used the placeholder notes as a buffer for the model. We then feed each following prediction into a queue to be used in the following inputs. There are 4 different predictions for each timestamp due to the structure of the output, which are each 7-length vectors. We sum each of these prediction vectors together, normalizing the result. This result is treated as a probability distribution, which we sample from for a final prediction for each timestamp. We treat each result this way rather than taking the sample from only the most recent prediction in order to bias the system towards longer term structure. This promotes the model to not just select the "most likely object," and avoid discounting underrepresented classes which otherwise arise from the inherent class imbalance in the dataset.

These predictions are then fed into a post-processing script which converts them into a playable format. This post-processing step eliminates "double positives" by locating notes that are only 23ms apart. Notes are almost never this close together in human charts, so we remove the later of the two notes when this occurs, which helps to smooth out the resulting output chart.

4 EVALUATION

Our goal is to create a chart generation model that leads to output with better patterning, defined as congruent combinations of notes. Deliberate patterning according to a given song is a task which requires expertise for human chart creators [1]. The task of patterning was divided into the separate tasks of onset detection and game object selection in prior work [6, 21, 22]. In our approach, we handled these two tasks within the same model with the goal of improving on patterning in comparison to prior works. After a brief overview of our evaluation methodology, we will go into detail regarding our baselines and the metrics we used.

The basic task of the evaluation is the generation of charts for withheld songs. We generate charts for these withheld songs using both our model and two other baselines, Dance Dance Convolution [6] and random noise. These baselines are elaborated on below. We convert all the charts into a binary format for comparison. This conversion involved stepping through each song 23ms at a time, representing any timestamp with an object that requires a discrete input with a 1, and everything else with a 0. We defined these discrete inputs to be any drum hit or beginning of a held object in Taiko, and any step or beginning of a held step in DDR. The purpose of this conversion was to make the comparison between the two models as fair as possible, as well as to allow cross-game comparisons.

The baseline we chose from prior chart generation work involving ML is Donahue's Dance Dance Convolution (abbreviated DDC) [6]. We selected DDC for three major reasons: DDC's ubiquity as a baseline, the similarities in game domain between Taiko and Dance Dance Revolution (abbreviated DDR), and access to similar quality human-authored charts for comparison.

DDC is a ubiquitous baseline for comparison within other prior PCGML chart generation works [21, 22]. DDC's general focus on

the task of chart generation provides the most clear baseline for comparison versus the two other mentioned prior works, which additionally focus on the separate tasks of sample classification [22] and improved onset detection methods [21]. Our specific focus is on evaluating patterning, which is unrelated to these separate tasks found in other work. Similarly, DDR and Taiko are non-keysounded games, which make them better candidates for comparison than Beatmania IIDX, the keysounded game domain of GenerationMania [22]. This is due to the significant differences between charting for keysounded and non-keysounded games which arise from the inherent restrictions imposed by keysounding [1]. Finally, we have access to a user-curated database of charts for DDR similar in form and function to our source of Taiko charts [25]. This allows us to find human charts for the same songs in both Taiko and DDR with appropriate difficulty levels for comparison that were reviewed and approved to be apart of a community database.

After collecting an evaluation set of 10 human-authored charts for the same songs in both Taiko and DDR, we generated charts for each song using TaikoNation and DDC respectively. It was necessary to collect separate human datasets for both Taiko and DDR due to differences in charting paradigms within each game domain. We will be comparing each chart generation method to each of the human-authored datasets individually. We convert all of the charts to the same binary representation described above to account for mild discrepancies in game objects between DDR and Taiko. We decided to select 10 charts for the evaluation set so that we could ensure that we covered an appropriate breadth of musical genres while ensuring both charts were of the appropriate difficulties and lengths to be fairly compared. In the selection of the Taiko charts, we used the same difficulty threshold we identified in Section 3.2 for selecting charts for the dataset, employing a similar threshold for DDR charts.

Along with DDC, we also selected random noise as a comparison baseline to be used as a control. We constructed this random noise by randomly choosing either 0 or 1 at each 23ms timestamp in the binary representation. This is used to evaluate how much structure both models are imposing on their output in comparison to human created charts. Noise also serves as a control for our chosen evaluation metrics, helping to justify that they are valid methods of measurement.

We make use of 5 metrics in our evaluation which involve comparisons against human charts to measure onset detection and patterning. These metrics are:

- Direct Comparison against Random (**DCRand**)
  Compares the binary representation of the given charts against random noise, counting the one-to-one similarities at each timestamp. This metric measures the AI output's distance from randomness to determine how much structure each approach is introducing to its output.
- Direct Comparison against Human (**DCHuman**)
  A one-to-one measure of similarities per timestamp between the binary representations of the AI generated and human authored charts for a given song. We chose this metric to give an overall picture of similarity between the given charts without a strict focus on onset detection. It is equivalent to accuracy, if we consider the human chart to be the gold standard.
- Onset Comparison w/ Scale against Human (**OCHuman**)
  A one-to-one measure of similarities as in DCHuman, with an added leniency window for detecting notes. For each note requiring input in the human chart, if there is no corresponding note in the AI chart at that exact timestamp, we check one timestamp ahead of and behind the note. This metric provides a broader focus on onset detection, with the leniency window accounting for mild timing discrepancies (±23ms).
- Overall Pattern Space (**Over. P-Space**)
  Compares the number of unique patterns present in the generated chart versus the overall potential space of possible patterns. We measure patterns by using a sliding window over 8 time stamps (8 x 23ms), counting each unique ordered combination of 8 that appears. We use this metric to determine how much of the potential pattern space each model is covering within its output, giving a sense of novelty found within the output. This metric gives us insight into how much of the overall possibility space each model is covering. This is important because our goal is to create a model that is able to represent the large amount of variance found within Taiko charts by different authors.
- Human Intersection Pattern Space (**HI P-Space**)
  Takes the intersection between the Over. P-Space of the model's chart vs the human's chart. This metric uses the same sliding window definition of patterning as above. The size of this set is compared to the size of the total set of human patterns in order to gauge what percentage of the "human" patterns the models are using.

With this combination of metrics, we aim to measure the performance of the models in terms of both patterning and onset detection.

5 RESULTS

The results of the evaluation outlined in Section 4 can be found in Tables 1 & 2. Table 1 contains the metrics measuring patterning, and Table 2 contains the metrics pertaining to onset detection. We also include Table 3 for additional context on how the human-authored charts performed on relevant metrics. Table 4 and Figure 5 provide information on the distribution of different note types found in the charts from the evaluation dataset. We review these results in detail below.

As shown in Table 1, our approach (TaikoNation) uses significantly more human patterns than DDC when evaluated using both the DDR and Taiko datasets. This means that TaikoNation is selecting many more patterns that are present in the human datasets. In comparison, DDC is using a smaller set of patterns consistently. A visual comparison between the two models demonstrating this difference can be seen in Figure 4, where each note is represented by a red line. In TaikoNation's output, the density between groupings of lines is much more varied, which is closer to both the human charts for DDR and Taiko. Comparatively, DDC's output has a much more consistent density, which reflects our observation above. The Overall Pattern Space (Over P-Space) for our approach is also much larger than DDC's, suggesting a larger variety of unique patterns

	Over. P-Space	HI P-Space DDR	HI P-Space T
DDC	15.938%	78.700%	83.160%
TN	**21.328%**	**92.470%**	**94.117%**

Table 1: DDC and TaikoNation compared on patterning metrics. Human Taiko dataset is abbreviated as T, and human Dance Dance Revolution dataset as DDR. Percentages are averaged from performance on all 10 charts in the dataset.

	DCRand	DCHuman DDR	DCHuman T	OCHuman DDR	OCHuman T
Random	//	50.185%	50.182%	65.580%	66.077%
DDC	49.938%	**76.430%**	**77.900%**	**80.900%**	**83.45%**
TN	**50.405%**	74.920%	74.987%	79.200%	79.323%

Table 2: Random, DDC, and TaikoNation compared on onset detection metrics. Human Taiko dataset is abbreviated as T, and human Dance Dance Revolution dataset as DDR. Percentages are averaged from performance on all 10 charts in the dataset.

	DCRand	Over. P-Space
Human DDR	**50.326%**	**16.055%**
Human Taiko	50.170%	14.453%

Table 3: Human DDR and Human Taiko datasets compared against relevant metrics. Percentages are averaged over performance on all 10 songs in the evaluation set.

	No Note	S Don	S Kat	B Don	B Kat	Roll	Denden
Human Taiko	82.180%	7.394%	7.305%	0.265%	0.310%	0.097%	2.449%
TaikoNation	83.270%	7.174%	6.934%	0.289%	0.347%	0.155%	1.830%

Table 4: Distributions of notes types found in the charts for the evaluation set of human-authored and TaikoNation generated Taiko charts. The definition of each note type can be found in Section 3.2.

can appear during chart generation. We hypothesize this is partially due to the variability present in the charts in our training dataset, which were all hand-authored by separate individual authors.

As can be seen in Table 2, TaikoNation's increase of patterning did not lead to a substantial increase in similarity to random noise. We interpret this to mean that our approach has learned more distinct human patterns without sacrificing the structure found in human charts. Notably, both DDC and our approach performed roughly equivalently to the human datasets on the DCRand metric, as per Table 3. Likewise, our performance on the onset detection metrics was not substantially different from DDC's. This indicates that our lack of a dedicated onset detection pipeline did not hamper our approach's onset detection ability in comparison to DDC.

Notably, we are unable to compare DDC and TaikoNation across the human datasets in terms of note type selection due to the differences in game domain. Instead we provide Figure 5 and Table 4 to give insight into TaikoNation's note type selection. Figure 5 shows a boxplot of TaikoNation's activations throughout the model's predictions while generating the charts for the evaluation set. These distributions are notably wider for more common note types such as "no note," and more sparse for less common note types such as "drumroll." We hypothesize this is due to the inherent class imbalance present in the dataset. Notably, the boxplots imply that the model has picked up on the structure of one of the uncommon note types, the "denden." The outliers in the denden boxplot may be occurring due to the denden's continuous structure, as it is much more likely that there is a denden object following a previous denden. TaikoNation's ability to recognize and properly use the denden sequentially may imply the model is also capable of recreating other structural patterns of note types. TaikoNation's ability to recognize and properly use the denden sequentially reinforces our earlier findings in Table 1 that suggest TaikoNation is using other patterns seen in human charts. Table 4 shows that TaikoNation's generated charts have a similar distribution of each note type to the human Taiko charts in the evaluation set. In combination with TaikoNation's performance on the patterning metrics seen in Table 1, we note that TaikoNation is using both similar rhythmic structures as well as similar amounts of each note type when compared to human-authored charts.

Overall, our approach used more patterns found in human-authored charts than DDC without a substantial difference in onset detection metrics. We hypothesize this is mainly due to the sliding window used for predictions, as well as the specifically curated dataset of high difficulty charts. These results support our initial claim that our approach leads to more congruent, human-like patterning than seen in prior chart generation works. However, we still need to confirm this with a human subject study, as our metrics are just an approximation of human judgement of patterning. We leave this for future work.

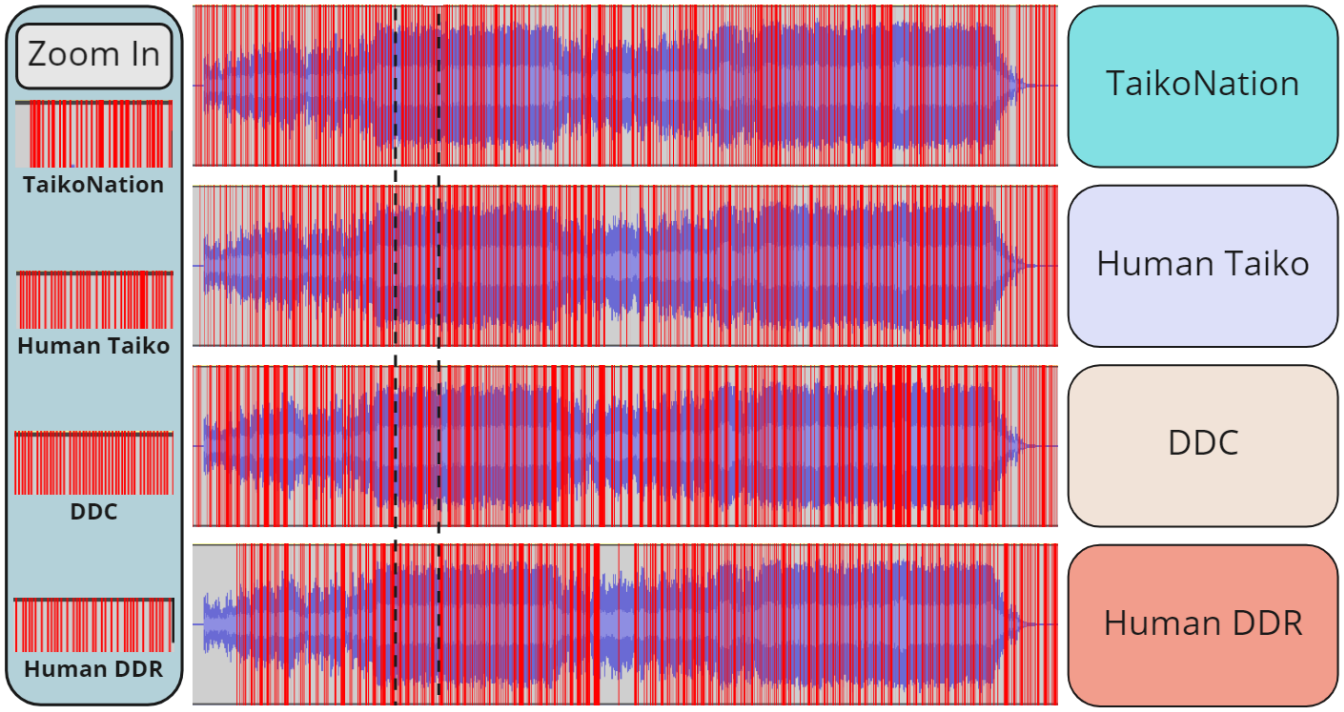

Figure 4: Model Comparison on the song *Nhato - Beyond the Seven*. Left window displays a close up look at the song segment outlined in black. Each red line represents a note object.

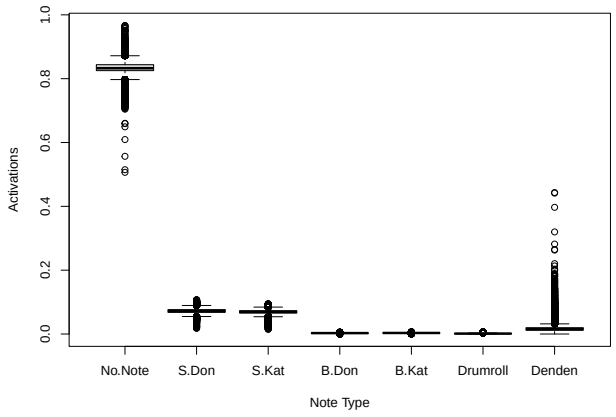

Figure 5: Boxplot depicting TaikoNation's predictions over the evaluation set. The definition of each note type can be found in Section 3.2.

6 LIMITATIONS AND FUTURE WORK

6.1 Technical Limitations

There are a few technical limitations within our approach that could be addressed to potentially improve our results. Our initial training approach lacked robustness. A possible solution could be training for a longer period of time with a custom stopping condition to avoid both overfitting and weight explosion. For example, we could train until the model does not predict at least one instance of every class on a validation set to avoid under-represented classes from being ignored. The training dataset we used was relatively small due to our focus of recreating human-like patterns found in the highest quality charts. A possible future approach could involve training on a larger set of charts of varying qualities as a baseline, then finetuning the model with a hand-selected smaller set of higher quality charts. Another limitation that stemmed from our focused dataset is the lack of difficulty control in our model which is present in DDC [6]. This was an intentional choice, as we chose to only use charts above a certain difficulty threshold in order to capture the patterning present in high difficulty charts. It may also be worthwhile to curate a dataset created by a single author to theoretically create a more consistent model, but there are very few chart creators who are prolific enough to do so. We did not implement more sophisticated onset detection techniques, such as the "fuzzy labelling" used in PCGoRG [21]. Implementation of these techniques within our generation pipeline could improve the onset detection of our approach. While our division of input audio into slices of 23ms is appropriate for the vast majority of songs found in rhythm games, the system's onset detection abilities are still limited by this choice for certain edge cases. In addition, though our pattern recognition and replication was shown to be relatively strong, they could be potentially improved through the use of architectures which have not been implemented in prior chart generation work.

For example, Transformers have been shown to perform well on sequence generation tasks [7], and could be employed in future attempts at chart generation.

6.2 Fundamental Limitations

While our initial findings show that we are performing well on the task of patterning, our metrics can only provide an approximation. In order to ascertain the value of the model, we need to conduct a human subject study to verify our findings. There are two major types of studies that we could perform: Designer-focused and Player-focused. Designer-focused studies would involve a group of chart creators working with our model and other baselines in a co-creative context [42]. We imagine a co-creative study akin to that of Guzdial et al. [12], in which chart creators interact with our model in a mixed-initiative fashion. Player focused studies would revolve around the comparison of human-authored charts with charts generated by our approach and other AI/ML baselines. Players would be given both types of charts to play in a blind setting, rating specific aspects such as patterning and rhythm choices from each. These studies would give us more insight into how well the model is learning to recreate human patterning in practice.

6.3 Future Work

The main hope for the future of this project is to incorporate our approach into a co-creative tool to be used by chart creators. This tool would ideally act as a teacher to novice creators learning to chart, and a powerful tool for more experienced creators to use at their own discretion. Building a notion of controllability into the model, meaning users could pick and choose desired aspects of the output, would be conducive to this goal. An avenue for future work with this model could be an ablation study examining which of the new contributions were most relevant to increasing the patterning ability of TaikoNation in comparison to prior work. This would be beneficial for future chart generation work, which could incorporate and enhance the most effective aspects of TaikoNation. While a full ablation study was out of scope for this paper, we performed a non-exhaustive ablation study while developing the final training approach and model architecture. The final approach used in this paper was settled on by qualitatively examining the output of the model after each change, selecting the version that produced the "best" patterning by our judgement. Exploring other game domains is another possibility for future work. There are many rhythm games with unique game objects and interesting limitations that would make the task of chart generation a difficult challenge. For example, Sound Voltex is a rhythm game with two continuous inputs in the form of knobs which can be twisted in various ways to remix the playable songs. This would require an approach that accounted for the keysounding and patterning of continuous inputs, which have yet to be explored. An interesting challenge could be the development of a general architecture that could be tweaked and modified per the requirements of a particular rhythm game domain.

7 CONCLUSIONS

Creating a chart generation model that replicates human-like patterning has proven to be a challenging, multifaceted task. We established a new approach for chart generation that produced charts with more congruent, human-like patterning than seen in prior work. This was shown through comparisons against a leading baseline across a number of pattern-focused and onset detection metrics. We also introduced a curated osu!Taiko dataset presented in a novel format, which could be used for a number of onset detection and chart generation tasks. We hope to help spur future chart generation work with a stronger focus on patterning, and aid novice and expert chart creators in the chart generation process.

ACKNOWLEDGMENTS

This work was funded through a Natural Sciences and Engineering Research Council of Canada (NSERC) Undergraduate Student Research Award (USRA).

REFERENCES

[1] 5argon. 2018. Game design and notecharting. https://exceed7.com/native-audio/rhythm-game-crash-course/game-design-and-notecharting.html
[2] Jont B Allen and Lawrence R Rabiner. 1977. A unified approach to short-time Fourier analysis and synthesis. *Proc. IEEE* 65, 11 (1977), 1558–1564.
[3] Dmitry Bogdanov, Nicolas Wack, Emilia Gómez Gutiérrez, Sankalp Gulati, Herrera Boyer, Oscar Mayor, Gerard Roma Trepat, Justin Salamon, José Ricardo Zapata González, Xavier Serra, et al. 2013. Essentia: An audio analysis library for music information retrieval. In *Britto A, Gouyon F, Dixon S, editors. 14th Conference of the International Society for Music Information Retrieval (ISMIR); 2013 Nov 4-8; Curitiba, Brazil.[place unknown]: ISMIR; 2013. p. 493-8.* International Society for Music Information Retrieval (ISMIR).
[4] Clayton Bonigut et al. 2020. .osu (file format). https://github.com/ppy/osu-wiki/blob/89c022f40cb9f8c3b59e2b43853ccf41fb84b471/wiki/osu!_File_Formats/Osu_(file_format)/en.md
[5] Clayton Bonigut et al. 2020. osu!taiko ranking criteria. https://github.com/ppy/osu-wiki/blob/89c022f40cb9f8c3b59e2b43853ccf41fb84b471/wiki/Ranking_Criteria/osu!taiko/en.md
[6] Chris Donahue, Zachary C Lipton, and Julian McAuley. 2017. Dance dance convolution. In *International conference on machine learning*. PMLR, 1039–1048.
[7] Linhao Dong, Shuang Xu, and Bo Xu. 2018. Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 5884–5888.
[8] Dylan Fitterer and Macedo CamachoPedro. 2008. AudioSurf.
[9] Cold Beam Games. 2011. Beat Hazard. http://www.coldbeamgames.com/beat-hazard.html
[10] Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber. 2002. Learning precise timing with LSTM recurrent networks. *Journal of machine learning research* 3, Aug (2002), 115–143.
[11] Nicholas Gillian, Sile O'Modhrain, and Georg Essl. 2009. Scratch-Off: A Gesture Based Mobile Music Game with Tactile Feedback.. In *NIME*. Citeseer, 308–311.
[12] Matthew Guzdial, Nicholas Liao, and Mark Riedl. 2018. Co-creative level design via machine learning. *arXiv preprint arXiv:1809.09420* (2018).
[13] Philippe Hamel, Yoshua Bengio, and Douglas Eck. 2012. Building musically-relevant audio features through multiple timescale representations. (2012).
[14] Dean Herbert et al. 2007. beatmap listing: osu! https://osu.ppy.sh/beatmapsets
[15] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
[16] Brandon Johannsen. 2016. osu!mapping tutorials. https://docs.google.com/spreadsheets/d/10Z5kWe084MQVHUcuZ5VATwz-ciIpt89XLmrCjibwyLw/edit#gid=0
[17] Annika Jordan, Dimitri Scheftelowitsch, Jan Lahni, Jannic Hartwecker, Matthias Kuchem, Mirko Walter-Huber, Nils Vortmeier, Tim Delbrügger, Ümit Güler, Igor Vatolkin, et al. 2012. Beatthebeat music-based procedural content generation in a mobile game. In *2012 IEEE conference on computational intelligence and games (CIG)*. IEEE, 320–327.
[18] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[19] Joseph Knoop. 2016. The Gigantic List Of All Rock Band 4's Available DLC Tracks. https://www.gameinformer.com/b/features/archive/2016/02/12/all-the-rock-band-4-dlc-in-one-list.aspx

[20] Honglak Lee, Peter Pham, Yan Largman, and Andrew Ng. 2009. Unsupervised feature learning for audio classification using convolutional deep belief networks. *Advances in neural information processing systems* 22 (2009), 1096–1104.

[21] Yubin Liang, Wanxiang Li, and Kokolo Ikeda. 2019. Procedural content generation of rhythm games using deep learning methods. In *Joint International Conference on Entertainment Computing and Serious Games*. Springer, 134–145.

[22] Zhiyu Lin, Kyle Xiao, and Mark Riedl. 2019. Generationmania: Learning to semantically choreograph. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, Vol. 15. 52–58.

[23] Roland Memisevic, Christopher Zach, Marc Pollefeys, and Geoffrey E Hinton. 2010. Gated softmax classification. *Advances in neural information processing systems* 23 (2010), 1603–1611.

[24] Adam Finn Nogaj. 2005. *A genetic algorithm for determining optimal step patterns in Dance Dance Revolution.* Technical Report. Technical report, State University of New York at Fredonia.

[25] Okami et al. 2021. ITG Packs Release Spreadsheet. http://itgpacks.com/

[26] Karl O'Keeffe. 2003. Dancing monkeys. *Masters project* (2003), 1–66.

[27] Anurag Sarkar and Seth Cooper. 2020. Sequential segment-based level generation and blending using variational autoencoders. In *International Conference on the Foundations of Digital Games*. 1–9.

[28] Jan Schlüter and Sebastian Böck. 2014. Improved musical onset detection with convolutional neural networks. In *2014 ieee international conference on acoustics, speech and signal processing (icassp)*. IEEE, 6979–6983.

[29] Jacob Schrum, Jake Gutierrez, Vanessa Volz, Jialin Liu, Simon Lucas, and Sebastian Risi. 2020. Interactive evolution and exploration within latent level-design space of generative adversarial networks. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*. 148–156.

[30] Gillian Smith, Alexei Othenin-Girard, Jim Whitehead, and Noah Wardrip-Fruin. 2012. PCG-based game design: creating Endless Web. In *Proceedings of the International Conference on the Foundations of Digital Games*. 188–195.

[31] Gillian Smith, Mike Treanor, Jim Whitehead, and Michael Mateas. 2009. Rhythm-based level generation for 2D platformers. In *Proceedings of the 4th international Conference on Foundations of Digital Games*. 175–182.

[32] Sam Snodgrass and Santiago Ontanón. 2016. Controllable Procedural Content Generation via Constrained Multi-Dimensional Markov Chain Sampling.. In *IJCAI*. 780–786.

[33] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.

[34] Stanley Smith Stevens, John Volkmann, and Edwin Broomell Newman. 1937. A scale for the measurement of the psychological magnitude pitch. *The journal of the acoustical society of america* 8, 3 (1937), 185–190.

[35] Adam Summerville and Michael Mateas. 2016. Mystical tutor: A magic: The gathering design assistant via denoising sequence-to-sequence learning. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, Vol. 12.

[36] Adam Summerville and Michael Mateas. 2016. Super Mario as a String: Platformer Level Generation Via LSTMs. In *Proceedings of the First International Conference of DiGRA and FDG*.

[37] Adam Summerville, Sam Snodgrass, Matthew Guzdial, Christoffer Holmgård, Amy K. Hoover, Aaron Isaksen, Andy Nealen, and J. Togelius. 2018. Procedural Content Generation via Machine Learning (PCGML). *IEEE Transactions on Games* 10 (2018), 257–270.

[38] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *arXiv preprint arXiv:1409.3215* (2014).

[39] Sarjak Thakkar, Changxing Cao, Lifan Wang, Tae Jong Choi, and Julian Togelius. 2019. Autoencoder and evolutionary algorithm for level generation in lode runner. In *2019 IEEE Conference on Games (CoG)*. IEEE, 1–4.

[40] TPGPL, Semro, and MegaApplePi. 2020. osu!taiko. https://osu.ppy.sh/wiki/en/Game_mode/osu!taiko

[41] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. 2015. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853* (2015).

[42] Georgios N Yannakakis, Antonios Liapis, and Constantine Alexopoulos. 2014. Mixed-initiative co-creativity. (2014).