AITG: A Machine Learning Approach to ITG Difficulty Analysis

Jace M. Williams

Indiana State University

GH401: Honors Independent Study

August 1, 2023

AITG: A Machine Learning Approach to ITG Difficulty Analysis

Stepmania is a free, desktop four-panel dance game inspired by *Dance Dance Revolution* (DDR) and its now-defunct American counterpart *In The Groove*. The latter of the two played the largest role in shaping the community as it stands today. For the purposes of this project, I will be referring to the game and the Stepmania open-source program as *In The Groove* (ITG).

The first iterations of *In The Groove*'s software utilized a block difficulty rating system similar to that of older DDR titles. In this system, charts were assigned a numerical rating, typically from 1 through 10 according to their perceived difficulty, by the game developer. Over the past couple decades following the end of ITG's official releases, the game's community has expanded upon that premise through open-source software, such as Stepmania, capable of running player-created content. Much of today's custom ITG content entails "Technical Charts" (Tech, for short). As Tech charts utilize novel patterns requiring more complex movements for proper execution, content quickly exceeded the traditional guidelines of a 1 through 10 difficulty scale. Herein lies the problem to be addressed by this Artificial Intelligence (AI) program.

As players created more difficult Tech charts, they adopted an expanded scale. In its current state, this scale typically runs up to a difficulty of 15, however a large majority of the difficult content played by the community lies in the small segment of 10 through 14. This provides very little granularity in distinguishing chart difficulties. Charts often may be perceived as noticeably different from one another despite falling within the same difficulty block number. When coupled with the fact that such perception is derived from a very intricate set of traits for any given chart, the numerical difficulty system begins to lack objectivity and precision in depicting a chart's difficulty.

Method

In order to develop an Artificial Intelligence model capable of providing a consistent and realistic difficulty rating for ITG Tech charts, a high quality training dataset must be selected. To this end, I selected three packs from a collection of open-source, user-created charts: "7Guys1Pack", "ITL Online 2023", and "Rumble In The Prairie 13 Singles". My choice for these particular packs was influenced by community consensus regarding the user-defined difficulty values in them, which provides a more objective target for extrapolated predictions based on them. Additionally, these packs have served as the grounds for competitive ITG tournament play, and therefore have a high degree of relevance to the community.

After collecting the raw dataset, the files for each chart were batched and preprocessed using the Simfile[1] Python library. The note data for each chart was iteratively stored in a two-dimensional array structure representing an ordered sequence. Each hittable note, either left, down, up, right, or a combination thereof, was depicted as a row entry in this matrix. Row entries are saved as a feature vector of length 10, representing the following 10 characteristics of the note: the elapsed song time, time since the previous note was played, a multilabel binary encoding of the note's direction(s), and a multilabel binary encoding of notes needing to be held down simultaneously.

Once the chart data from the three selected packs was preprocessed, subsets of this collection of note sequences were selected as the training and validation datasets for an AI model. Next, TensorFlow's Keras Functional API[2] was used to compile a sequence of layers into

[1] https://github.com/garcia/simfile
[2] https://keras.io/guides/functional_api/

a model object which could be fit to the user-defined outputs of the training dataset, and then utilized for difficulty prediction of unknown inputs based on the layer weights calculated during training. The primary method of experimentation used in the model's development was the selection, ordering, and tuning of the layers used when compiling the model.

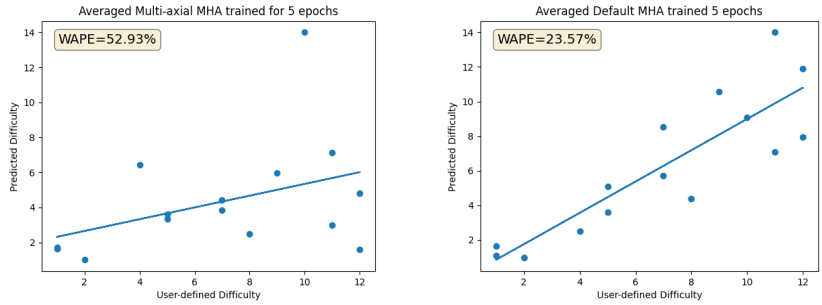
Results

The design of the model utilized a Multi-Headed Attention[3] (MHA) layer, which is a machine learning feature typically used in language sequence decoding and translation. In this case, the layer was used in a Self-Attention[4] method to identify features of a chart's note sequence most relevant to its perceived difficulty. These attention weights were then averaged across the pool to converge on a single numerical weight value representing the chart. Finally, the discrete chart difficulty values were normalized across the range of values observed in the training dataset to generate practical predictions. A Weighted Average Percent Error (WAPE) was used to evaluate the accuracy of predictions as compared to the values given to the charts by their authors.

In the first testing run of this model, I used the "7Guys1Pack" set of ITG Tech charts as the training input, and reserved a subset of the charts for prediction validation. The MHA layer

[3] https://keras.io/api/layers/attention_layers/multi_head_attention/
[4] https://theaisummer.com/self-attention/

was configured to analyze both the row (notes) and column (note features) axes.

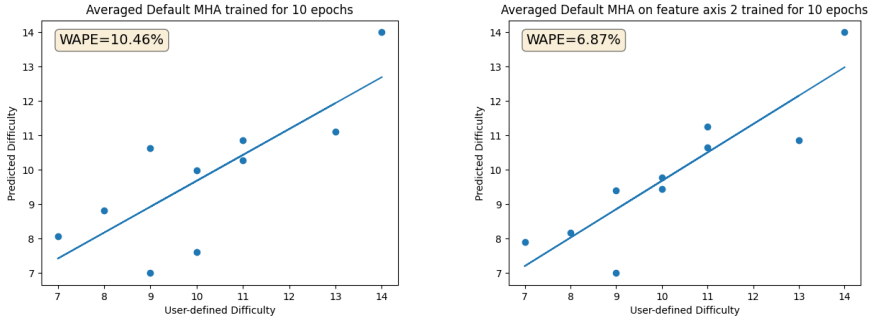


As seen by the line of best fit in the first graph, the MHA approach using both axes tended to greatly under-rate chart difficulty, and had an unacceptable WAPE of ~53%. Using the default single-axis rowwise setting of the MHA layer, the model began more appropriately rating charts on the higher end of the user-defined scale. One weakness still displayed by the model at this point, however, was an unnecessarily strong correlation between chart length (number of notes) and the predicted difficulty, rather than utilizing the more subtle features and patterns of the data itself. One possible cause of this issue may have been over-fitting of the training data.

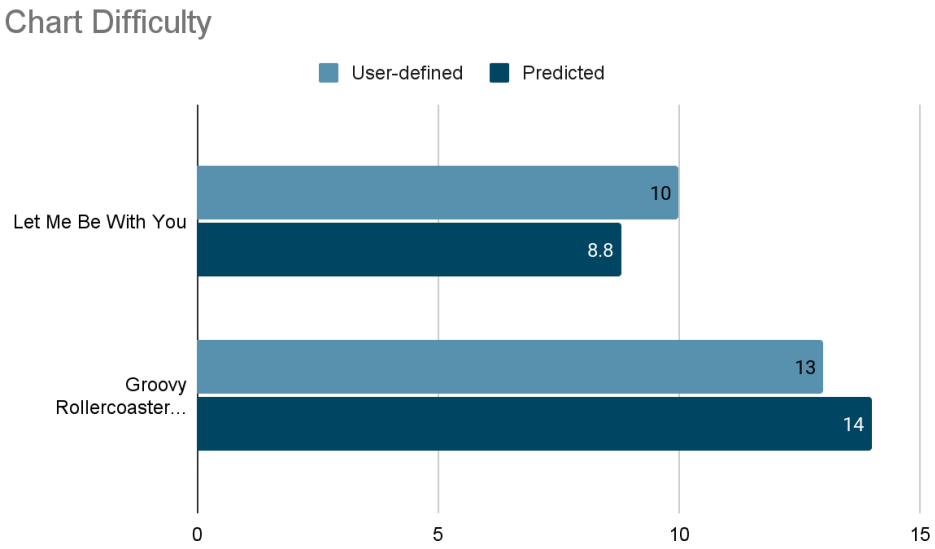
In order to alleviate the over-fitting issue, further training was carried out using a Huber Loss[5] function and optimized with an exponential decay scheduled learning rate. Compared to the standard loss metric of sparse categorical crossentropy and gradient descent optimizer, these settings allowed for a more gradual and generalized adjustment of weights over a longer training duration. In addition to these training changes, the dataset used for training and validation was refined. It was found that the "7Guys1Pack" collection possessed unusual and somewhat impractical user-defined ratings of charts on the lower end of the spectrum. Going forward, I opted to rely on a combination of the "ITL Online 2023" and "Rumble in the Prairie 13"

[5] https://doi.org/10.1214/aoms/1177703732

collections for training and validation. As seen in the following graphs, these changes provided a substantial improvement to the alignment of prediction difficulties with their user-defined counterparts. In the right-hand graph, the MHA layer of the model was adjusted to attend to the column axis of the chart data, representing the individual note features. This change provided further repeatable improvement to the model's predictions and was subsequently used for chart difficulty evaluation for the ITG Technical community.



Two "ITL Online 2023" charts which held significant community interest related to perceived difficulty were "Let Me Be With You" and "Groovy Rollercoaster Acid Trip", originally rated at 10 and 13 by their respective authors. Many disagreed with these human evaluations, however, deeming the 10 to be an overestimation and the 13 to be too conservative a rating in the context of the event. Extrapolating from the rest of the "ITL" dataset, the model was used to provide updated predictions for the charts' difficulties:



These predictions were then reported to the community, with overall feedback being that the programmatically generated values from the model aligned more closely with perceived difficulty than the user-defined values did.