Generating Japanese Pop Music by Building On MusicGen

Dihong Huang
Georgia Tech
dhuang329@gatech.edu

Akaash Dash
Georgia Tech
adash37@gatech.edu

Saikanam Siam
Georgia Tech
siam@gatech.edu

Abstract

In this study, we address the challenge of fine-tuning an artificial intelligence-based music generation model, MusicGen, to generate Japanese Pop (J-pop) music. Our approach involves enhancing MusicGen with a custom-curated J-pop dataset, applying transfer learning techniques to integrate genre-specific nuances into the pre-trained model. Our technical approach centers on two key phases: full fine-tuning and LoRA fine-tuning of MusicGen, focusing primarily on its autoregressive transformer-based decoder. To assess the efficacy of our approach, we employ quantitative metrics, namely Cross-Entropy Loss and Perplexity, alongside qualitative evaluations through human listening tests. The results from these evaluations indicate a successful adaptation of MusicGen to the J-pop genre, as evidenced by the model's enhanced ability to generate music that aligns with the structural and stylistic characteristics of J-pop. Notably, our experiments reveal that full fine-tuning on a comprehensive J-pop dataset leads to a significant reduction in Cross-Entropy Loss and Perplexity, suggesting an improved predictive accuracy of the model. However, the implementation of LoRA posed challenges due to the complex architecture of MusicGen and resource limitations. Also, the fine-tuned model exhibited a worsen performance in generating genre-specific music, as confirmed by subjective assessments from human listeners. The generated music captured some of the distinct style of J-pop but failed to maintain a high level of audio quality. The main reasons could be limitation of training data and training time, which leaves large room for future improvement when better GPUs are accessible for us. The findings of this study contribute to the growing field of AI-driven music generation, offering a not successful but still meaningful approach to model fine-tuning for specific music genres. Our work not only showcases the potential of AI in artistic creation but also sets the stage for future explorations in the intersection of technology and music, with implications in entertainment, therapy, and education.

1. Introduction

Music generation using artificial intelligence has made significant strides, allowing for the creation of intricate musical compositions that can emulate the nuance and creativity of human composition. However, the challenge often lies in fine-tuning these generative models to produce music that is not only high quality but also tailored to specific prompts or styles. This fine-tuning process is essential because, despite the versatility of pre-trained models, they may not always align with the unique requirements of specialized musical tasks.

The objective of this work is to refine the music generation capabilities of MusicGen [2], a state-of-the-art model, by fine-tuning it on a custom-curated J-pop composition dataset. The goal is to leverage transfer learning, a method by which a model developed for a task is repurposed as the starting point for a model on a second task. By initiating training from a pre-trained checkpoint, we aim to preserve the extensive knowledge the model has already acquired while integrating new, task-specific insights with a smaller subset of data.

To augment the fine-tuning process, we propose the application of Low-Rank Optimization for Approximate (LoRA) [6] techniques. These techniques are designed to enhance the model's memory efficiency by modifying the model's architecture, such as by incorporating low-rank adaptation matrices into the self-attention layers.

The significance of this approach lies in its potential to streamline the music generation process, tailoring outputs more closely to user inputs and reducing the computational resources required. In the landscape of AI-generated music, ensuring that the output matches the intended style or mood set by a prompt is crucial. This is not just a matter of technological curiosity, but of practical utility in fields ranging from entertainment to therapy. In music therapy, the ability to generate music tailored to individual preferences or therapeutic needs can significantly enhance the effectiveness of treatment. For instance, music that aligns with a patient's emotional state or therapeutic goals can provide a more personalized and impactful therapeutic experience. This is especially beneficial for individuals with mental health issues,

where music can serve as a non-invasive and engaging form of therapy. Moreover, the refined model could catalyze future studies by providing a robust framework for exploring the intersection of AI and creative arts. This could lead to a deeper understanding of how AI can aid in artistic expression and creativity, opening new avenues for collaboration between technologists and artists. Additionally, the educational implications are profound, as such advancements could be integrated into curriculum to teach students about the interplay between technology and arts, thereby fostering a new generation of interdisciplinary experts. Overall, this research has the potential to not only revolutionize the field of AI music generation but also to enrich human experiences and education in a meaningful way.

In summary, this work is poised to contribute to the expanding domain of AI-driven music creation by providing a methodologically sound and technologically innovative approach to model fine-tuning. The anticipated outcome is a more accurate alignment between the music generated and the textual prompts provided, thus enhancing the model's utility and efficiency.

2. Related Works

Our endeavor builds upon the considerable body of work in AI music generation. Earlier studies have demonstrated the feasibility of generating music with neural networks, such as the DeepBach [5] model tailored for polyphonic music of Bach's style. Further, OpenAI's Jukebox [3] presented a model capable of generating music in various genres, complete with lyrics and melody. However, our approach differs in its emphasis on fine-tuning to prompts and the introduction of LoRA to optimize for memory efficiency. Moreover, there is a methodological advancement in our intended creation of a tailored dataset, which addresses a gap in available resources for such specialized tasks.

MuLan presents a pioneering model in joint audio-text embedding, leveraging a massive dataset of music recordings and natural language annotations. This approach underscores the potential of integrating music information retrieval with natural language processing, an aspect central to our project's goals [7].

MusicLM, another significant work, introduces a model for generating high-fidelity music from text descriptions. This model, using a hierarchical sequence-to-sequence modeling approach, adheres closely to text prompts and transforms melodies based on textual input. The introduction of the MusicCaps dataset, comprising music-text pairs, is particularly instructive for our project's focus on dataset creation for text-to-music generation tasks [1].

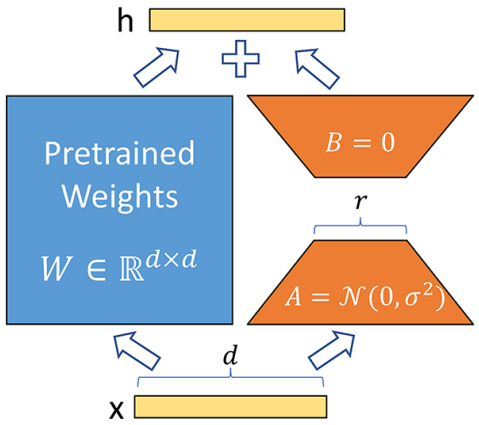
AudioGen, a textually guided audio generation model, operates on learned discrete audio representation. Its two-stage process, involving audio encoding and auto-regressive language modeling, aligns closely with our project's objectives. AudioGen's focus on high-fidelity audio generation and text adherence, along with its exploration of multi-stream modeling and classifier-free guidance, provides valuable insights for enhancing MusicGen's efficiency and text alignment [9].

Collectively, these works emphasize the importance of dataset quality, model architecture, and cross-modal learning in AI-driven music generation. They offer methodologies and insights critical to refining MusicGen for specialized tasks like Japanese pop music generation, underlining the significance of integrating cross-modal elements and advanced modeling techniques.

3. Technical Approach

Our method is fine tuning MusicGen on a specific dataset to achieve a more tailored music generation. This is based on transfer learning, by initializing the training on a pretrained checkpoint we retain most of the knowledge and add task specific knowledge to the model with less training data. In addition, we propose to apply Low-Rank Optimization for Approximate (LoRA) on this task to further improve memory efficiency.

Figure 1. How LoRA works



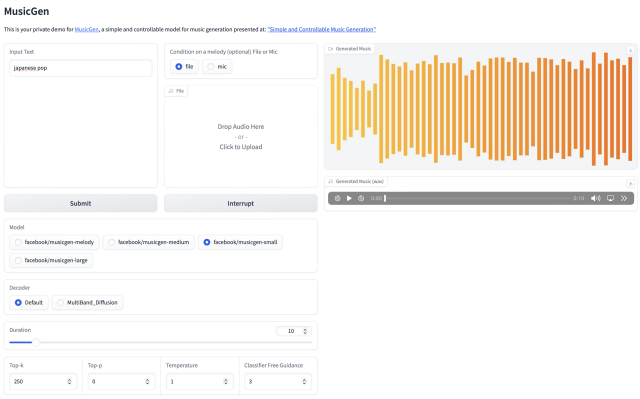
MusicGen is comprised of a single-stage transformer LM together with efficient token interleaving patterns. Its architecture can be divided into two main parts: a Encodec auto-encoder for audio tokenization and quantization and an autoregressive transformer-based decoder. We will focus on the decoder.

The implementation of our approach is in two stages: full fine tuning and LoRA fine tuning. Meta released the training code for MusicGen, which greatly facilitates our coding for full fine tuning. In contrast, LoRA fine tuning may require change in original model architecture and source code, such as replacing the self attention layer with a pair of low-rank adaption matrices.

The baseline will be the pretrained model, in our case

the MusicGen 32khz small model. It can be easily accessed either through the official demo or API call.

Figure 2. MusicGen Demo



3.1. Evaluation and Loss Functions

For evaluation, we planned to employ a two-sided approach: quantitative metrics and qualitative human judgment. We wanted to compare the performance of the fully fine-tuned model and the LoRA-enhanced model against the baseline pre-trained MusicGen model using cross-entropy loss, Frechet Audio Distance [8], and Contrastive Language-Audio Pretraining metrics [4]. Subsequently, human listeners would conduct ear tests to provide subjective assessments of the audio quality.

We ended up using Cross Entropy Loss and Perplexity. Cross Entropy Loss measures the difference between the probability distributions of the actual and the predicted sequences. In our context, it helps in quantifying how well the model predicts the next token in a sequence of music. Lower cross-entropy loss values indicate that the model's predictions are closer to the actual data, suggesting better performance. Perplexity is a measure of how well a probability distribution predicts a sample. In our case, it assesses the model's ability to predict the sequence of musical tokens. A lower perplexity score corresponds to a model making more accurate predictions, indicating a better understanding of the music structure and style.

We did not end up employing Frechet Audio Distance (FAD) and Contrastive Language-Audio Pretraining (CLAP) metrics. The environment setup was complicated not well supported by MusicGen. During our experiment, there were frequent errors in loading pretrained checkpoint for CLAP and disalignment with the format of our training data. In addition, running these evaluation metrics were computationally heavy and time consuming, even Meta disabled them in their default configuration.

3.1.1 Cross Entropy Loss

Cross-entropy loss, a fundamental concept in the realm of machine learning, particularly in classification tasks, plays a pivotal role in evaluating the performance of generative models like MusicGen. At its core, cross-entropy loss is a measure of the difference between two probability distributions - the predicted probability distribution output by the model and the actual distribution represented by the true data. In the context of music generation, where the task is to predict the next sequence of musical tokens, this metric quantifies the effectiveness of the model in capturing the underlying patterns and structures of the musical genre.

The computation of cross-entropy loss involves the application of the following formula: $-\sum_i y_i \log(p_i)$, where y_i is the true label in one-hot encoded form and p_i is the model's predicted probability for that class. For a well-performing model, the predicted probabilities should align closely with the true distribution, resulting in a lower cross-entropy value. Conversely, a high cross-entropy value indicates a disparity between the model's predictions and the actual data, signaling poor model performance.

In our project, we employed cross-entropy loss as a key metric to evaluate and fine-tune the MusicGen model for J-pop music generation. By analyzing the cross-entropy loss, we could gauge how closely the model's predictions of the next musical token aligned with the actual sequence in the dataset. This metric was crucial for iteratively refining the model during the training process. By monitoring changes in cross-entropy loss, we were able to make informed decisions about model adjustments, such as tweaking hyperparameters or modifying the model architecture, to enhance its predictive accuracy and, consequently, its ability to generate musically coherent and stylistically aligned compositions.

The choice of cross-entropy loss as an evaluation metric was underpinned by its ability to provide a clear, quantitative measure of the model's performance. This was particularly important in the context of our project, where the goal was not only to generate music but to do so in a way that is stylistically and thematically consistent with the specific nuances of the J-pop genre. The cross-entropy loss, therefore, served as an essential tool in our endeavor to push the boundaries of AI-driven music generation, ensuring that the output of the MusicGen model was not only of high quality but also resonant with the intricate styles of Japanese pop music.

3.1.2 Perplexity

Perplexity is a widely used metric in natural language processing and, by extension, in sequence generation tasks such as music generation, where the objective is to predict a sequence of tokens (in this case, musical notes or elements). Formally, perplexity is a measure of how well a probability

model predicts a sample. It is mathematically defined as the exponential of the entropy of the probability distribution, or equivalently, as the inverse probability of the test set, normalized by the number of words.

In the context of our project, perplexity served as a crucial metric for evaluating the performance of the fine-tuned MusicGen model. The model, akin to language models, generates sequences of musical elements. Perplexity in this scenario quantifies the model's effectiveness in predicting these sequences. Specifically, a lower perplexity score indicates that the model's predictions are more closely aligned with the actual sequences in the test dataset, implying a better understanding and replication of the musical structure and style inherent in the data.

The calculation of perplexity in our project follows the standard approach used in language modeling. It involves computing the likelihood of the sequence of tokens generated by the model, given the actual sequence in the test set. This likelihood is typically calculated using the cross-entropy loss between the predicted and actual distributions over the token sequences. The perplexity is then obtained by taking the exponential of this cross-entropy loss. The formula is given by:

$$\text{Perplexity} = e^{(\frac{-1}{N}\sum_{i=1}^{N}\log(P(w_i)))}$$

where N is the number of tokens in the test set, $P(w_i)$ is the probability of the i-th token predicted by the model, and the summation is over all tokens in the test set.

In our project, we leveraged perplexity to assess how well the fine-tuned MusicGen model adapted to the specific stylistic and structural nuances of J-pop music. A lower perplexity score indicated that the model was effectively learning and predicting the patterns and sequences characteristic of J-pop, which was essential for achieving our goal of generating music that closely aligns with the given prompts and the stylistic elements of the genre. Thus, perplexity was not only a measure of the model's predictive accuracy but also an indicator of its capacity to capture the essence of J-pop music.

4. Data

4.1. Motivation for Dataset Creation

The dataset was compiled to fine-tune the Musicgen model. Our goal is to enable the model to learn diverse J-pop styles from a set of popular Japanese artists and hence generate similar music. The data is specifically intended for research and development in the field of AI-driven music generation. It is aimed at enhancing the model's ability to generate novel, high-quality music tracks. The dataset is designed for machine learning practitioners and researchers focusing on music generation and related AI applications in the field of digital audio processing.

4.2. Composition of the Dataset

The dataset comprises two primary types of data: music clips and metadata. The audio clips are the J-pop songs splited into 30-second wav files, and the metadata is provided in jsonl format. There are two jsonl files handling the meta data for train and test set, each containing lines of dictionaries corresponding to each music clip in that dataset. For a specific music clip "Sawano Hiroyuki Sh0ut - chunk150.wav", the meta data is a dictionary formatted like this:

Metadata	**Value**
key	F#
artist	Sawano Hiroyuki
sample rate	44100
file extension	wav
description	Electronic, Industrial, ... Synth-pop song
duration	30.0 s
bpm	188
genre	Electronic, ... Synth-pop
title	Sawano_Hiroyuki_Sheut...158.wav
instrument	synthesizer, drums, ... drummachine
moods	epic, energetic, ... love
path	/content/drive/MyDrive/...chunk150.wav

Table 1. Example format of a music clip

The dataset contains more than 7000 30-second music clips and hence more than 7000 dictionaries describing them. The songs comes from 29 J-pop singers/groups/bands, some are ranked most popular in 2023 and some are personal favorites. They covers a broad spectrum of styles within J-pop to see if Musicgen can learn diverse patterns within J-pop – a large genres containing different subgenres and mix.

4.3. Data Collection Process

In the first phase of data collection, we intended to utilize the spotify_scraper.py script, which interfaced with the Spotify Web API. This script was instrumental in collecting detailed metadata and audio features of J-pop tracks. The Spotify API offers extensive data about tracks, including but not limited to the track's name, artist, album information, release date, and popularity score. More importantly, it provides audio features like tempo, key, valence, and energy, which are critical in understanding the musical attributes of each track. However, we encountered rate limit of Spotify API and it was less convenient than direct download from youtube. Therefore, we switched to youtube and found alternative for the audio attributes for labeling.

The complete data collection and processing pipeline consists of three stages: download, label, and adjustment. We applied a Python library yt_dlp to download the songs

from given youtube playlists. We picked a playlist with roughly 20-40 songs for each artist we chose on youtube. The music clips are required to be in 30 second duration by the model, we then applied splitting and dividing them into train and test dataset. The labeling part was automatically done with a library called essentia for audio analysis. We loaded the files with librosa and set up the TensorflowPredict2D model from essentia to extract genres, moods, and instruments of these songs. All of these labels were compiled into one dictionary per clip and written into the corresponding json files. In fact, Meta prompted their training data manually with professional musicians. Although we were aware of the effectiveness and granuality of manual label, it was both time and money costly to do so.

4.4. Maintenance of the Dataset and Limitation

The dataset will be hosted in Google Drive and may be updated to include new music clips and metadata by interests. There would not be version control since it is intended for personal usage. In terms of limitation, this dataset is only for training on J-pop and contains very limited amount of songs and artists.

5. Experiments and Results

5.1. First Stage with Smaller Dataset

In the first stage of this project, we used full fine tuning to train MusicGen on a small playlist consisting of 26 songs by Yoasobi for code debugging and preliminary results.

The training process took roughly 3 hours on one T4 GPU on Google Colab for 3 epochs. Meta used a different definition of epoch here: one epoch refers to the smallest amount of computation that we want to work with before checkpointing. With update_per_epoch set to 2000 step, we effectively run the model for 6000 steps. Here is the train and validation summary of CE and perplexity.

Data	Time	LR	Grad Norm	Grad Scale	Cross-Entropy (ce)	PPL
200/2000	0.71 it/sec	2.28E-05	2.471E+00	19807.522	3.565	40.718
400/2000	0.74 it/sec	1.86E-05	2.436E+00	32768.000	3.449	37.391
600/2000	0.75 it/sec	1.47E-05	2.503E+00	32768.000	3.508	38.015
800/2000	0.76 it/sec	1.12E-05	2.498E+00	32768.000	3.562	39.101
1000/2000	0.76 it/sec	8.10E-06	2.423E+00	32768.000	3.364	34.233
1200/2000	0.76 it/sec	5.48E-06	2.454E+00	32768.000	3.408	34.957
1400/2000	0.77 it/sec	3.35E-06	2.465E+00	32768.000	3.419	34.767
1600/2000	0.77 it/sec	1.73E-06	2.452E+00	32768.000	3.364	32.880
1800/2000	0.77 it/sec	6.38E-07	2.476E+00	32768.000	3.446	35.394

Table 2. Training progress table for smaller dataset, Epoch 3

From the CE and perplexity we could see that the model is clearly overfitted with our small dataset. Both CE and PPL are not stable and bump up and down. With roughly 200 audio clips being run 6000 steps, each sample was visited by the model 30 times on average. According to our ear test, the generated clips certainly captured the style of Yoasobi: catchy melody and abundant usage of synthesizer and keyboard. However, the model performed poorly when

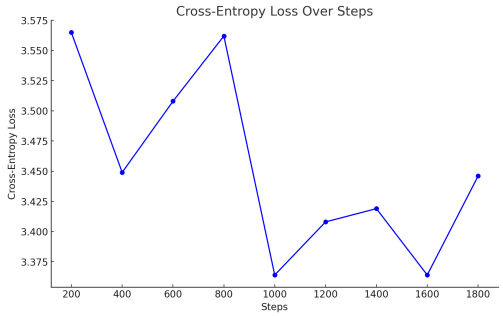


Figure 3. Cross-Entropy Loss Over Steps for smaller dataset

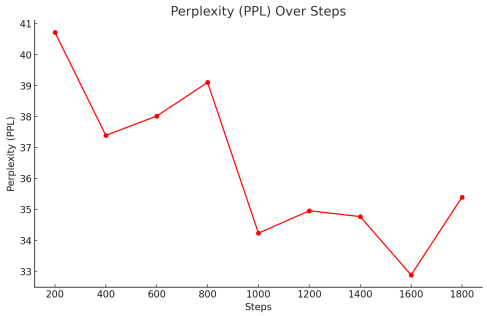


Figure 4. Perplexity (PPL) Over Steps for smaller dataset

asked to generate other genres of music, like country songs or heavy metal. It is likely that the model weights were too heavily tuned toward specialiazing on Yoasobi music that lots of pretrained knowledge got destroyed.

5.2. Second Stage with Larger Dataset

5.2.1 Failed Attempt of LoRA

After collecting the full dataset, we first attempted to conduct LoRA on MusicGen. Meta engineered the model with different implementation of cross attention from their own research-based library XFormer and created complicated model structure, which prevented us from modifying the code efficiently. Moreover, Meta warned that they currently don't support fine tuning a model with slightly different layers, unless a custom checkpoint file is crafted. It turned out to be extremely less feasible within our ability and limited computational resources. As a result, we decided to forfeit the application of LoRA in this project and focused on obtaining a satisfying results from full fine-tuning.

5.3. Full Fine-Tuning on Larger Dataset

Due to the size of model parameter, it is impossible to fit training loop more than 5 epochs, in total 10000 steps on a single T4 GPU. Also, it was infeasible to do extensive hyperparameter tuning with limited computation resources. With experiences of previous tests and advice from a expert,

we set up the configs optimizing performance on a single GPU while remaining accuracy. The configs we used are shown below:

Parameter	Value
solver	musicgen/musicgen_base_32khz
model/lm/model_scale	small
continue_from	//pretrained/facebook/musicgen-small
continue_from	/content/drive/MyDrive/finalth.checkpoint3
conditioner	text2music
dset	audio/train
dataset.num_workers	2
dataset.valid.num_samples	1
dataset.batch_size	2
schedule.cosine.warmup	8
optim.optimizer	adamw
optim.lr	1e-4
optim.epochs	3
optim.updates_per_epoch	2000
optim.adam.weight_decay	0.01

Table 3. Configuration Parameters

The solver is a training process design by Meta, and we chose to apply the one for small MusicGen_ base_32khz model. The two continue_from were used to initiate the training from pretrained checkpoint and last checkpoint we saved. The conditioner is text2music rather than melody since we wanted to train a text-to-music model. The others are for dataloader and trainer setup, such as batch_size, lr_scheduler and optimizer.

We split the training process into several 3 epoch runs to prevent checkpoint loss due to Colab disconnect or memory limitations. By specifying continue_from to the saved checkpoint from last run, we could resume the training without starting from scratch. To make sure the model visit each sample at least twice, we set epochs = 9 and trained for three runs. We didn't keep the results for first two run, and the results of last run (epoch 7 - epoch 9) are shown below:

Data	Time	LR	Grad Norm	Grad Scale	Cross-Entropy (ce)	PPL
200/2000	0.83 it/sec	2.28E-05	2.106E+00	262144.000	4.302	83.217
400/2000	0.83 it/sec	1.86E-05	INF	207093.760	4.287	81.442
600/2000	0.83 it/sec	1.47E-05	2.024E+00	131072.000	4.267	79.955
800/2000	0.83 it/sec	1.12E-05	2.012E+00	131072.000	4.355	84.485
1000/2000	0.83 it/sec	8.10E-06	1.969E+00	131072.000	4.244	77.961
1200/2000	0.83 it/sec	5.48E-06	1.977E+00	131072.000	4.209	76.317
1400/2000	0.83 it/sec	3.35E-06	1.952E+00	131072.000	4.197	76.176
1600/2000	0.83 it/sec	1.73E-06	1.962E+00	131072.000	4.224	75.668
1800/2000	0.83 it/sec	6.38E-07	1.958E+00	131072.000	4.306	82.588

Table 4. Training progress table for larger dataset, Epoch 9

The metrics for full data training are also unstable, but the model is obviously not overfitting. In fact, we found the model somewhat undertrained during ear tests. We randomly selected 10 description from the test prompts and asked both our fine-tuned model and the baseline to generated 15-second clips correspondingly. 7 out of 10 times we assessed the baseline outperformed our model. Some of the clips generated by our model are bad and more like

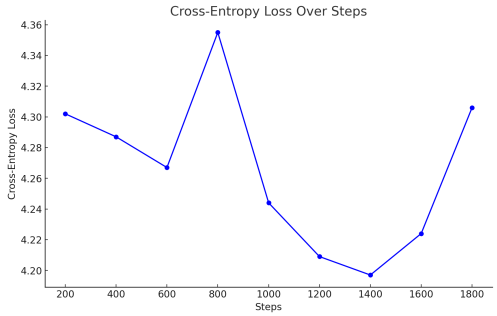


Figure 5. Cross-Entropy Loss Over Steps for larger dataset

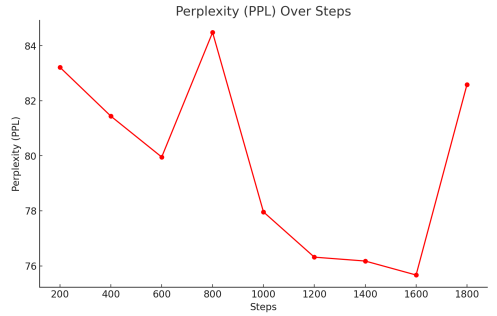


Figure 6. Perplexity (PPL) Over Steps for larger dataset

random noise. It was disappointing to see this, but we were able to identify some potential causes for our model failure: insufficient training data and training time, poor labels, and perhaps low quality audios (the music extracted from youtube videos may not be compatible to professionally recorded audios), etc. Nonetheless, we successfully built a working end to end fine-tuning pipeline for MusicGen, which could be updated and further improved to create a better fine-tuned MusicGen.

5.4. Failed Attempt of GPT enhanced prompts

In the first stage, we considered to use ChatGPT to enhance our prompts for training, by asking it to extend the labels we input and output a fine-grained description with some variations. However, the GPU consumption of calling OpenAI API was unexpectedly high especially when we already had a large model running. It would be a good trick to try if we have access to much better GPUs and could possibly improve quality of generated music.

6. Conclusion

In conclusion, our research in fine-tuning the MusicGen model for generating Japanese Pop (J-pop) music presents significant advancements in the field of AI-driven music generation. The application of a custom-curated J-pop dataset, coupled with transfer learning techniques, has resulted in the successful adaptation of MusicGen to produce

music that embodies the stylistic and structural characteristics of J-pop.

The quantitative assessments, primarily Cross-Entropy Loss and Perplexity, have revealed that full fine-tuning on a comprehensive J-pop dataset significantly improved the model's predictive accuracy. This is evident from the reduced values in both metrics, suggesting that the model has become more adept at generating sequences that closely align with the actual music structures in J-pop. However, the challenges encountered in implementing Low-Rank Optimization for Approximate (LoRA) fine-tuning, due to MusicGen's complex architecture and resource constraints, limited the exploration in this aspect. Unfortunately, the fine-tuned model did not demonstrate notable improvements in producing genre-specific music, which was also corroborated by qualitative assessments from human listeners. The reasons could be limited training data, poor labeling, inadequate training time, or inefficiency.

Future work in this area can explore several avenues. Firstly, overcoming the technical and resource limitations to successfully implement LoRA fine-tuning could further enhance the model's efficiency and effectiveness. Additionally, expanding the dataset to include a wider variety of J-pop styles and sub-genres could help in developing a more versatile and robust model. Another potential area of research is the exploration of how different types of textual prompts, possibly augmented with advanced tools like GPT-4, can influence the quality and diversity of the generated music. This could pave the way for more personalized and contextually relevant music generation.

The implications of our work extend beyond the realm of AI and music generation. By demonstrating the potential of AI in artistic creation, this research opens new possibilities in entertainment, therapeutic applications, and educational settings. Despite of our bad results, this project showcases the intersection of technology and art, providing valuable insights into the future of creative AI applications. The attempts made in this project not only contribute to the enrichment of AI-generated music but also offer a glimpse into the exciting possibilities that lie at the intersection of technology, creativity, and human experience.

References

[1] Andrea Agostinelli, Timo I. Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, Matt Sharifi, Neil Zeghidour, and Christian Frank. Musiclm: Generating music from text, 2023. 2

[2] Jade Copet, Felix Kreuk, Itai Gat, Tal Remez, David Kant, Gabriel Synnaeve, Yossi Adi, and Alexandre Défossez. Simple and controllable music generation, 2023. 1

[3] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music, 2020. 2

[4] Benjamin Elizalde, Soham Deshmukh, Mahmoud Al Ismail, and Huaming Wang. Clap: Learning audio concepts from natural language supervision, 2022. 3

[5] Gaëtan Hadjeres, François Pachet, and Frank Nielsen. Deepbach: a steerable model for bach chorales generation, 2017. 2

[6] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. 1

[7] Qingqing Huang, Aren Jansen, Joonseok Lee, Ravi Ganti, Judith Yue Li, and Daniel P. W. Ellis. Mulan: A joint embedding of music audio and natural language, 2022. 2

[8] Kevin Kilgour, Mauricio Zuluaga, Dominik Roblek, and Matthew Sharifi. Fréchet audio distance: A metric for evaluating music enhancement algorithms, 2019. 3

[9] Felix Kreuk, Gabriel Synnaeve, Adam Polyak, Uriel Singer, Alexandre Défossez, Jade Copet, Devi Parikh, Yaniv Taigman, and Yossi Adi. Audiogen: Textually guided audio generation, 2023. 2

Name	**Contribution**
Dihong Huang	Data processing, training and technical implementation
Akaash Dash	Data collection, theoretical background
Saikanam Siam	Data processing and loss functions

Table 5. Contribution Table