Generating Japanese Pop Music by Fine Tuning MusicGen

Dihong Huang
Georgia Tech
dhuang329@gatech.edu

Akaash Dash
Georgia Tech
adash37@gatech.edu

Saikanam Siam
Georgia Tech
siam@gatech.edu

Abstract

This project introduces a effective approach to enhancing Japanese pop music generation by fine-tuning the MusicGen model through transfer learning and Low-Rank Optimization for Approximate (LoRA) techniques. The fine-tuning is conducted in two stages: set up the training pipeline for fully fine tune with MusicGen's training code, followed by LoRA adjustments to the model architecture for improved memory efficiency. The effectiveness of the approach is measured against the baseline MusicGen 32kHz small model using cross entropy loss, Frechet Audio Distance, and Contrastive Language-Audio Pretraining metrics, complemented by human subjective evaluation. To facilitate the fine-tuning process, we are compiling a unique dataset of 30-second audio clips with corresponding descriptive prompts, leveraging the Spotify API for metadata to assist in labeling. This method aims to achieve more precise prompt-aligned music generation, addressing the gap in specialized datasets and setting a new standard for the quality and applicability of AI-generated music.

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 [1], 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) [5] 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.

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 [4] model tailored for polyphonic music of Bach's style. Further, OpenAI's Jukebox [2] 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.

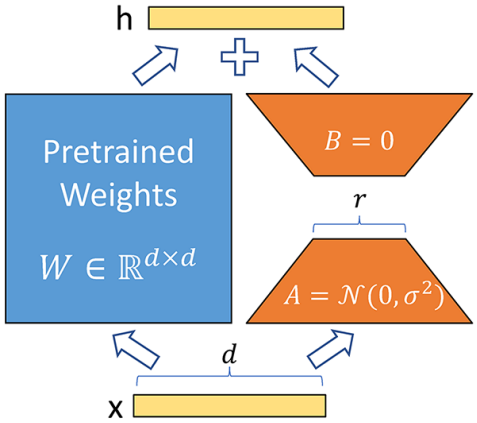
In summary, this work is poised to contribute to the expanding domain of AI-driven music creation by providing a methodologically sound and technologically innovative approach to model fine-tuning. The anticipated outcome is a more accurate alignment between the music generated and the textual prompts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

2. 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 pre-

trained 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

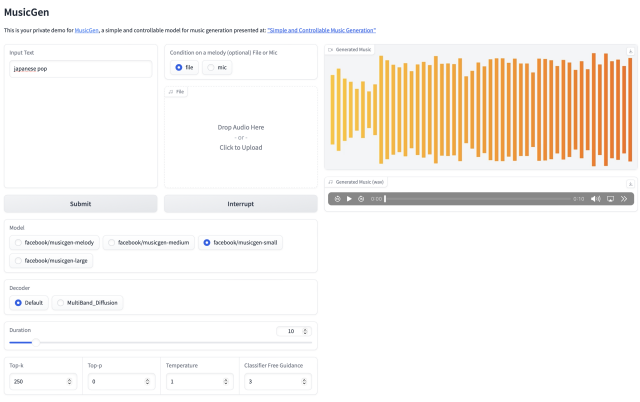


MusicGen is comprised of a single-stage transformer LM together with efficient token interleaving patterns. Its architecture can be divided into two main parts: a Encodec auto-encoder for audio tokenization and quantization and an autoregressive transformer-based decoder. We will focus on the decoder

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



For evaluation, the project will employ a two-sided approach: quantitative metrics and qualitative human judgment. We will 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 [6], and Contrastive Language-Audio Pretraining metrics [3]. Subsequently, human listeners will conduct ear tests to provide subjective assessments of the audio quality.

3. Preliminary Results

The dataset we intend to use consists of 30s audio clips paired with corresponding prompt texts that describe the musical content and attributes they contain. Currently, there is no such dataset available so we will have to manually collect, label, and split it. To help with the labeling process, we decided to download music directly from Spotify as Spotify API also provides metadata about audio attributes like bpm, acousticness, etc. We also need to split each song into 30s chunks for the training process. The data collection is still work in progress, but we already have the complete data processing pipeline built and tested.

Since the full training data is not ready, 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 playlist can be found here:
https://www.youtube.com/playlist?list=PL8ByKNaSDmbF6KuBIwrTeFd94aoNjhr
The training process took roughly 3 hours on one T4 GPU on Google Colab for 3 epochs. Here is the train and validation summary of CE and perplexity.

Figure 3. Train and Valid Summary

(a) Train Summary

```
Train | Epoch 3 | 200/2000 | 0.71 it/sec | lr 2.28E-05 | grad_norm 2.471E+00 | grad_scale 19807.522 | ce 3.565 | ppl 40.718
Train | Epoch 3 | 400/2000 | 0.74 it/sec | lr 1.86E-05 | grad_norm 2.436E+00 | grad_scale 32768.000 | ce 3.449 | ppl 37.391
Train | Epoch 3 | 600/2000 | 0.75 it/sec | lr 1.47E-05 | grad_norm 2.503E+00 | grad_scale 32768.000 | ce 3.508 | ppl 38.015
Train | Epoch 3 | 800/2000 | 0.76 it/sec | lr 1.12E-05 | grad_norm 2.498E+00 | grad_scale 32768.000 | ce 3.562 | ppl 39.101
Train | Epoch 3 | 1000/2000 | 0.76 it/sec | lr 8.10E-06 | grad_norm 2.423E+00 | grad_scale 32768.000 | ce 3.364 | ppl 34.233
Train | Epoch 3 | 1200/2000 | 0.76 it/sec | lr 5.48E-06 | grad_norm 2.454E+00 | grad_scale 32768.000 | ce 3.408 | ppl 34.957
Train | Epoch 3 | 1400/2000 | 0.77 it/sec | lr 3.35E-06 | grad_norm 2.465E+00 | grad_scale 32768.000 | ce 3.419 | ppl 34.767
Train | Epoch 3 | 1600/2000 | 0.77 it/sec | lr 1.73E-06 | grad_norm 2.452E+00 | grad_scale 32768.000 | ce 3.364 | ppl 32.880
Train | Epoch 3 | 1800/2000 | 0.77 it/sec | lr 6.38E-07 | grad_norm 2.476E+00 | grad_scale 32768.000 | ce 3.446 | ppl 35.394
Train Summary | Epoch 3 | lr=8.68E-06 | grad_norm=2.468E+00 | grad_scale=31465.472 | ce=3.452 | ppl=36.200 | duration=2602.837
```

(b) Valid Summary

```
Valid Summary | Epoch 3 | ce=4.453 | ppl=85.843 | duration=33.799
```

From the CE and perplexity we could see that the model is clearly overfitted with our small dataset. The other two metrics are not yet implemented due to long run time and complex config and environment settings.

The LoRA implementation is much more complicated than we expected, as Meta used its own research package and lots of self defined classes for the model. Meta also warned that they currently don't support fine tuning a model with slightly different layers. We will continue working on it, but it may not be feasible within our ability.

There is also another potential augmentation to our text prompts for training. Due to time and fund constraint, we

are not able to manually write expert captions to the songs. We applied Spotify audio attributes as a baseline prompts, but it is possible to use other tools like ChatGPT to augment these prompts. By calling OpenAI API, the baseline prompts will be input into GPT4 or GPT3.5 to get an enhanced version. The code snippet to do so might look like this:

```
if use_gpt4:
  response = openai.ChatCompletion.create(
      model="gpt-4",
      messages=[
          {"role": "system", "content": sys_prompt},
          {"role": "user", "content": simple_prompt},
      ],
      temperature=1.0,
  )
else:
  # fallback to gpt3.5 if user does not have gpt4 api access
  response = openai.ChatCompletion.create(
      model="gpt-3.5-turbo",
      messages=[
          {"role": "system", "content": sys_prompt},
          {"role": "user", "content": simple_prompt},
      ],
      temperature=1.0,
  )

out_prompts = response.choices[0]['message']['content'].strip()
print('Generated prompts:')
```

We will test how this can affect runtime of the training process and quality of generated audios in the following stage of development.

link for video: https://drive.google.com/file/d/1P65LcwR-f2ANhPd3ujiaxrAmDExFrQd9/view?usp=sharing

References

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

[2] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music, 2020. 1

[3] Benjamin Elizalde, Soham Deshmukh, Mahmoud Al Ismail, and Huaming Wang. Clap: Learning audio concepts from natural language supervision, 2022. 2

[4] Gaëtan Hadjeres, François Pachet, and Frank Nielsen. Deepbach: a steerable model for bach chorales generation, 2017. 1

[5] 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

[6] Kevin Kilgour, Mauricio Zuluaga, Dominik Roblek, and Matthew Sharifi. Fréchet audio distance: A metric for evaluating music enhancement algorithms, 2019. 2