MuSiC: Music Similarity Composer with Multi-Tasked Generative Models

Jack Michaels
Department of Computer Science, AI
Stanford University
Stanford, CA
jackfm@stanford.edu

GitHub Link: https://github.com/jmichaels32/cs330/tree/main/project

Abstract—**This paper explores music generation under the lens of a multi-tasked architecture. Specifically, we frame music generation as the multi-tasking of generative 'instrument players' where each generative head contributes the melody for a particular instrument. The current literature deviates from this; modern approaches like Meta's MusicGEN or OpenAI's Jukebox use EnCodec with highly time dependent Codebook interleaving patterns [1] or, alternatively, VQ-VAEs [2]. Our architecture differs significantly, exploring not only the viability of multi-tasked generative models in music generation, but also testing the viability of generative heads in a multi-tasked model.**

To ensure consistency within the same distribution, audio data was scraped from YouTube within one genre and from one artist. The selected genre was LoFi due its straightforward melodies, recognizable sound, lack of vocals, and consistent instruments. The artist chosen was "LoFi Girl - Chill Beats" [8] who was chosen for audio quality and data availability. To reduce computational requirements, audio samples were spliced into 30 second audio chunks where zero padding was used for audio chunks with length less than 30 seconds.

To train the generative heads representing instruments, each audio sample was separated into its most obvious instruments (bass, drums, piano, other) using the modern instrument separation tool spleeter [6]. To better support model diversity and prevent the generative heads from collapsing in on themselves (and learning the identity function), when a song is inputted into the model to generate a shared song 'inspiration' embedding, a randomly selected audio chunk was chosen from the dataset to act as labels for the generative heads.

This paper tests three models: MusicGEN conditioned only on audio as a baseline, MusicGEN conditioned on both text and audio as an alternative baseline, and a multi-tasked generative model. Our multi-tasked generative model uses a vision transformer as a shared embedding layer and the WaveGAN generative model as its heads. Both of these architectures take in raw song data provided by the YouTube repository described above. MusicGEN conditions on raw audio samples while the inputs to our model use MEL spectrograms to convert the raw audio signal into the frequency domain and into an image (to thus apply the vision transformer). Both MusicGEN and our model output raw audio.

Our results indicate that while generative multi-tasked architectures may be promising, due to resource, time, and data constraints, our model is unable to outperform the baseline. MusicGEN is a state of the art model coming out of Meta AI, an institution with vast training resources and teams of machine learning engineers. Further testing and resources are largely required to not only empirically determine the effectiveness of generative heads in a multi-task framework, but the effectiveness of this architecture in relation to music generation. In other words, the scope of this project was unfortunately too large, though sufficient time has been spent developing functionality for the model.

Empirically, our three evaluation metrics are as follows: Fréchet Audio Distance (FAD) [7], average Mean Squared Error (MSE) and human survey. FAD compares statistics computed on a set of reconstructed music clips to background statistics computed on a larger set of gold labels in order to empirically test out-of-distribution-sounding generated music. MSE naively computes this, offering a metric on our deviation from the original gold label. And finally, a human survey determines how easily humans are able to differentiate between generated and gold label songs.

Both baseline MusicGEN models achieve an FAD score around 4 while our model achieves an FAD score of around 18, suggesting definitive randomness in our outputs. Furthermore, MusicGEN achieves an average MSE of around $4.2 * 10^8$ while our model achieves an MSE of around $1.98 * 10^9$, further supporting this claim. Finally, when conducting the human evaluation, every contestant was able to differentiate our model from the gold label.

I. Introduction

Following the modern accomplishments of AI in image generation and large language models, investigating AI in music generation has been ever more frequently explored. Given the human resources and time required to develop modern music by hand, machine learning alternatives and the involvement of AI in the development of music are reaching higher and higher demands. This can take the form of a music composition tool (like Google's Magenta [5]), but it may also directly take the form of music generation. This paper explores music generation under the lens of a multi-tasked architecture, specifically a multi-tasked architecture with generative heads.

This research was inspired to simulate the current music generation process: an artists is tasked with compositing multiple instruments, all of which have their own unique melodies and rhythms but all of which must remain thematically synonymous and on time. To do this, the artist doesn't generate all instruments at once but instead decides on a general ”direction” or ”emotion” that the song is trying to convey, and then composes said instruments separately to best suit that ”emotion”. However, the composition of these instruments is inherently interlinked: in other words, each instrument's generation will inform the others in this cyclical process. Note that this paper is only attempting to generate music 'similar' to another song, not generate music spontaneously from an inputted description or melodic pattern.

We explore music generation in a tailored environment by selecting Lofi as a genre due to its simple melodic nature and heavy reliance on standard instruments (with no vocals). Lofi is also a primary candidate for music generation because of its ready use as study music and lack of artistic expression/creativity. Due to the heavy time and computational constrains imposed, we kept this genre choice consistent throughout training.

Architecturally, we chose to explore two variants of MusicGEN as our baseline: MusicGEN only conditioned on audio and MusicGEN conditioned on both audio and text. This is in contrast to our multi-tasked generative model which uses a Vision Transformer as the shared embedding layer and WaveGAN for each generative head. Due to music's high dependency on frequencies (notes), we chose to represent our audio samples as mel spectrograms to provide the model with more informative audio representations (see Figure 1). Note that music generation is a considerably harder task than general human speech generation (Text To Speech/TTS) due to the expanded frequency domain of music in comparison to frequencies producible by humans. Worth mentioning in conjunction to this is the alternative to music generation: symbolic music generation. This alternative was considered, though due to the expressive limits of MIDI and symbolic formats, was not pursued.

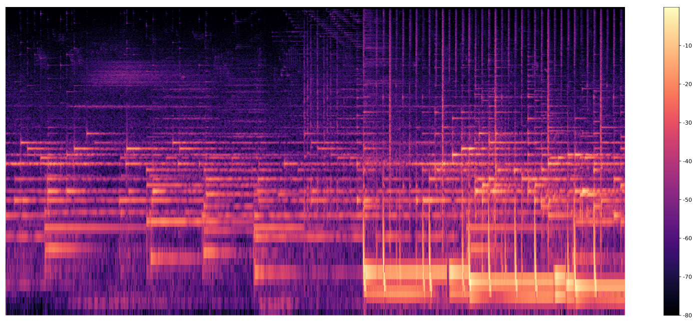

Fig. 1: Spectrogram for a 30 Second Audio Chunk

II. Related Literature

A. Modern Architectures in Music Generation

Due to the nuance and complexity of music generation, there are an extremely large number of explored architectures that have high nuance and limited success. Most applicable to this paper is MusicGEN [1] from Meta which uses EnCodec with time dependency by incorporating Codebook interleaving patterns to solve the task of music generation. This model achieves impressive success with this architecture, most notably the generated audio follows standard music music theory best-practices (like staying on beat). Alternative models like OpenAI's Jukebox [2] use VQ-VAEs to embed songs into a vector and then apply transformer based approaches to generate audio waveforms. An applicable architecture which has largely fallen out of flavor is WaveNet [10]. Though this architecture is dated (RNNs), its exploration into modelling the time dependency within music contributes significantly to this paper. This idea of time dependency and time constraints (like staying on beat) is reflected in nearly all papers and is an important aspect of creating an accurate music generating model.

Highly related to this paper are the contributions provided by WaveGAN [3] and Vision Transformers (ViT) [4] which outline seminal architectural explorations into expressive generative models that may pertain to music generation. Specifically, WaveGAN offers a modified generative adverserial network (GAN) designed specifically for audio generation. WaveGANs apply GANs directly to raw audio-waveform outputs (See Figure 2), designed for sound effect generation but nonetheless theoretically applicable to general music generation. Similar to these augmentations are ViTs, which provide explorations into a vision based transformer as opposed to the classic text based transformer. ViTs modify the query, values and keys into a image patch problem, thus allowing us to train a transformer adjacent architecture on image data. Since we generate mel spectrograms for each audio chunk, we can readily utilize this architecture.

Related to the architectural choices outlined above are the high dependencies on music embeddings which plague the field of music generation. Standard music 'embedding' software like MIDI formats are inherently inexpressive and severely handicap any model's ability to generate semantically significant music. An important player in this space are vggish [9] embeddings which generate expressive and statistically significant embeddings and are used in important evaluation metrics like the Fréchet Audio Distance described below.

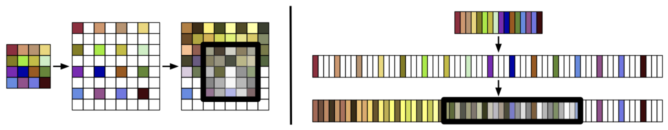
Fig. 2: Standard GAN Architecture **(left)** Compared to WaveGAN Architecture **(right)**

B. *Evaluation Metrics*

The evaluation metrics used in the literature are equally as diverse as the architectures used. Assessing music quality is a nuanced topic, and correspondingly there aren't many rigorously determined metrics that exist. Human evaluation (turing test) is the most common evaluation technique [11] [3] [1] [2]. Another metric which frequents music generation papers is the Fréchet Audio Distance (FAD) [11] [7] [1] which shared similarities between KL divergence and calculates a metric based on how dissimilar a song's local distribution is from the gold label song distribution. It is clear that more work needs to be done on creating an accurate music-quality evaluation metric.

III. Datasets

The dataset consists of 1170 Lofi songs with a sample rate of 32kHz, each approximately two minutes long taken from a playlist on the YouTube channel "Lofi Girl - Chill Beats" [8]. This playlist was chosen due to its length, consistent audio quality, lack of instrumental diversity, and the fact that it all came from the same artist. All these factors are important as they guarantee the music comes from the same distribution, which, given the computational resource limitations, is required to better support convergence.

Extensive dataset preprocessing was required to prepare the data for the multi-tasked generative architecture (See Figure 3). This involved splicing all audio samples into 30 second chunks (where zero padding was used if audio samples were less than 30 seconds). This audio is sufficiently processed to be passed into the baseline architecture MusicGEN, however, more steps are required to process the data for use in the multi-tasked generative model. This involved two steps: generating mel spectrograms from these audio chunks as input into the model and generating the gold labels for the generative heads from a randomly selected song. Mel spectrograms were generated using the fast fourier transform while the gold labels were generated using a pretrained instrument separation model *spleeter* [6]. *Spleeter* separates instruments into five categories: vocal, drums, bass, piano, other. The vocal files generated were thrown out.

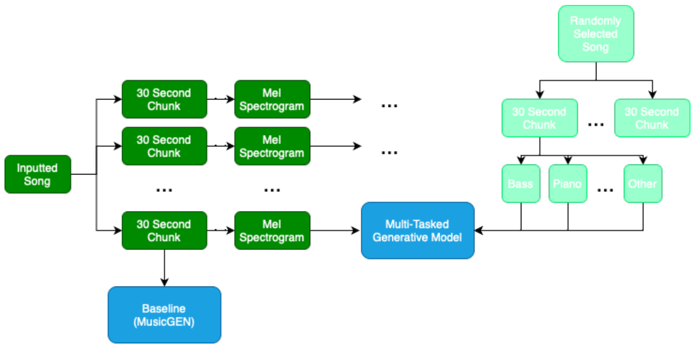

Fig. 3: Data Preprocessing

IV. Methods

A. Baseline

For our baseline, we chose to use the state of the art music generator to test the limits of our model. We used two variants of MusicGEN [1] as our baseline: MusicGEN only conditioned on audio and MusicGEN conditioned on both audio and text. MusicGEN operates by using an autoregressive transformer-based decoder conditioned on a text or melody representation. This reflects our architecture: we used a transformer based architecture to generate embeddings. MusicGEN deviates from this step onwards where it starts to take these embeddings and pass them into a compression model which uses Residual Vector Quanitization (RVQ). This leads to different parallel 'streams' which comprise of discrete tokens originating from different learned codebooks (See Figure 4).

B. Multi-Tasked Generative Model

As previously described, our model is a multi-tasked generative model with WaveGAN heads and

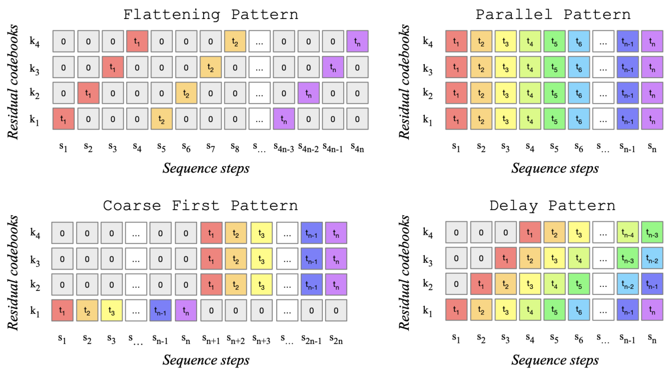

Fig. 4: Codebook Interleaving Patterns for MusicGEN

a ViT shared bottom layer. Inputs to the model consist of 30 second audio segments translated to mel spectrogram (See Figure 3). With a sample rate of 32kHz, 30 seconds of audio corresponds to a feature vector of size 960,000, demonstrating the sheer size of the input space. In standard digital music representations, these values can range from -32768 to 32768.

The model architecture is better described in Figure 5. The architecture takes in mel spectrograms and passes that into the vision transformer model described above. The hope is that the model will be able to learn not only temporal relations within the mel spectogram, but pick up on specific music theory patterns that it can exploit in its own music generation. From here, once we have generated a sufficient song embedding, we can pass this into the WaveGAN model to train a generator that will output music audio. In order to avoid collapse (and to avoid the algorithm needing to learn in its weights how to perform an inverse fourier transform/Griffin-Lim), we train the WaveGAN on another randomly selected song as to motivate the network to create 'similar sounding music'.

As previously described, the mel spectrogram is generated with a particular song. This is passed into the Vision Transformer in order to generate some sort of "song embedding" which can inform the WaveGAN architectures onto which direction they may need to take. Since WaveGAN architectures directly train on song data, we have to force the vision transformer to output a vector instead of a matrix representing the embedding of the song. Since embeddings don't convey the same meaning as song data, this may have ultimately decreased the productivity of our model. However, with that being said, upon sufficient training time it is hoped that the model should be able to learn this correspondence and ultimately continue generating high quality music.

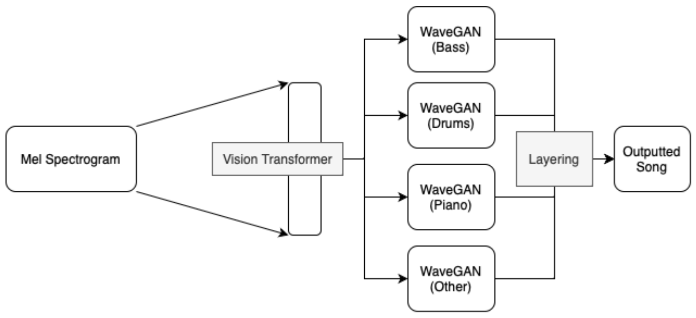

Fig. 5: Model Architecture

This architecture was trained for around six hours under Azure GPUs. The specific vision transformer used was vit_b_16 which corresponds to a sample rate of 16kHz and a 'basic' architecture. A learning rate of 0.001 was used for training the WaveGANs.

C. Evaluation Metrics

We chose three evaluation metrics: FAD score [7], MSE error, and a limited scale Turing test based human evaluation. FAD score uses the vggish model [9] to generate effective song embeddings. We generate these embeddings from both generate music and gold label music. KL divergence between these song embeddings is them calculated to determine the distributional difference between our generated songs and actual songs. The Fréchet Audio Distance between two Gaussians is given in Figure 6.

$$\mathbf{F}(\mathcal{N}_b, \mathcal{N}_e) = \|\mu_b - \mu_e\|^2 + tr(\Sigma_b + \Sigma_e - 2\sqrt{\Sigma_b \Sigma_e})$$

Fig. 6: FAD Between Two Gaussians

Naively, we then calculate standard Mean Squared Error (MSE) on generated songs with their true audio counterparts. This metric provides less information than one might assume because two frequencies can sound the same (be resonance of one another), but take different values. Thus, a song which is an octave higher may yield an extremely high MSE even though they are, to us, almost the same songs.

Finally, a Turing test type evaluation metric was used to determine which human preference. This was conducted by showing humans two songs: one generated song and the original, gold label song which was used to generate that similar song. They were then prompted to pick the "generated song". This metric allows us to discover which model humans prefer (the most effective evaluation metric), and which model is performing close to the actual supposed distribution. This evaluation metric is represented as the percentage of trials where generated music was incorrectly classified as 'not generated'. If a human was completely unsure which song is generated and which is not, we would have an accuracy of 50% on this metric. Due to a lack of time, human evaluation was limited to 50 tests.

V. Results

The section "Conclusive Results" outlines the 'conclusive'/quantitative results we obtained based on our evaluation metrics described previously. Since audio can't be represented on paper, the section will "General Results" will briefly describe the sound of the output from these models. If you would like to investigate more and listen for yourself, please visit the GitHub linked above.

A. Conclusive Results

Our results suggest that, while multi-tasked generative models are viable alternatives to modern architectural approaches like MusicGEN, more training resources are required to accurately access whether multi-tasked generative models are architecturally less advanced and less equipped than modern models. See Table I for results.

	FAD	MSE	HE
MusicGEN + music	4.8	4.1×10^8	8%
MusicGEN + text and music	3.3	4.2×10^8	8%
ViT + WaveGAN	15.5	1.98×10^9	0%

TABLE I: Evaluation Metrics for each Architecture (HE): Human Evaluation

All of these metrics support MusicGEN's ability to readily generate music from a conditioned song. Our architecture is barely able to generate conceivably sensible melodies as supported by the FAD score, MSE and human evaluation. Due to the high dimensionality of the architecture and the high variability in inputs, this projects training requirements seem to have gotten ahead of themselves. It is clear during training that more data is required and more efficient training techniques must be taken to effectively train this architecture. Results are generally inconclusive, primarily due to these training requirements not being met. With that being said, it is clear that modern music generation softwares like MusicGEN are advanced enough as they were able to fool a human 4 times.

B. General Results

Generally, upon inspection of the generated audio, we can made some connections to infer not only how the model is performing, but what directions should be taken to improve model success. Model outputs primarily sound like noise. Given how many parameters must be updated in this architecture and how variable each input is (mel spectrograms vary widely from audio chunk to audio chunk), it is reasonable to assume that convergence will take longer than allocated. Nonetheless, while initial results sounded like pure noise, after training there are brief periods of tonality shifts and signs of audio manipulations. These auditory artifacts happen without pattern, suggesting that the model has no ability to recognize the exact rhythm required for music composition. This correlation directly follows from the model's unconstrained inputs; it can generate any waveform it wants where if we enforced symbolic music generation then beat matching would directly fall into place.

Worth mentioning, the model consistently matches specific tones between the inputted songs and generated music. This is further supported by the fact that some generated songs have periods of relative silence where melodic patterns are not present. These higher level music composition 'extractions' suggest that the model has an extremely high-level view of music generation. It hypothesized that upon further training, these higher level functionalities will distill down into lower level detail. Ultimately, it is clear that more computation is required to accurately asses these models; though it is promising to hear the current results.

VI. Future Work

A. Resources

Due to the computational resources required to train a generative model, let alone a music generating model, there are a lot of avenues for future work on this paper considering that the majority of computational resources required for this project were not met. Nearly immediate improvements can be made based on the computational requirements; in other words, since the code base has already been developed, as long as we have more computational resources we will be able to steadily decrease loss on the model. This is in conjunction to increasing the data (or else we would reach early convergence).

On another note, while exploring this area we found a need for a better digital symbolic representation for music. MIDI formats are an inherent misrepresentation of the music they mean to portray due to a lack of nuance and variability. This nuance and variability is often what commonly makes music enjoyable. Due to the drastic reduction in state space once we start considering symbolic representations for music instead of the semi-continuous representation given by raw audio formats, we will start achieving not only better sounding music but more accurate and theoretically precise audio. However, this architecture has yet to be developed and is thus an area of open exploration.

B. Architectural Modifications

Architecturally, there are a lot of open questions posed by this research that are left to explore. While mel spectrograms, vision transformers, and WaveGANs were used to accomplish the multi-tasked generative architecture, it is possible to swap out these architectures with potentially more effective ones. Initial architectures which come to mind are diffusion models for the generative heads and a custom audio transformer for the shared layer. In the literature, audio transformers seem yet to be developed though we have already seem how transformer can be adapted to adjacent AI fields like vision.

Given the variability in song distribution, music generation/similar music composition could be viewed as its own task for each song. In this sense, we could apply black box meta learning on top of the multi-task model to better improve the performance on new queries.

Finally, our output only consists of 30 second chunks of audio (matching our inputted audio chunk size). These audio chunks do not have a comprehensive start and end, and modern music almost never lasts less than two minutes. Future work is required to stich together these audio samples into a comprehensive song.

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.

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

[3] Chris Donahue, Julian McAuley, and Miller Puckette. Adversarial audio synthesis, 2019.

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[5] Google, 2023.

[6] Romain Hennequin, Anis Khlif, Felix Voituret, and Manuel Moussallam. Spleeter: a fast and efficient music source separation tool with pre-trained models. *Journal of Open Source Software*, 5:2154, 06 2020.

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

[8] LofiGirl-ChillBeats, 2023.

[9] tensorflow, 2023.

[10] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio, 2016.

[11] Li-Chia Yang and Alexander Lerch. On the evaluation of generative models in music. *Neural Computing and Applications*, 32(9):4773–4784, May 2020.