Music Generation with Direct Policy Optimization

Shad Torrie, Dallin Miner, Christopher Kitras
Electrical and Computer Engineering Department, Brigham Young University, Provo, UT 84602, USA
{shad.torrie, keyofd, kchristm}@byu.edu

Abstract

Music is a fundamental creative outlet that allows humans to express themselves meaningfully. However, not every human who has the desire to express themselves through music has the skill set to do so effectively. MusicGen is a music generation model that allows anyone to input a text or audio prompt and receive a generated piece of music as output. In this paper, we present a system that allows a potential music composer to utilize MusicGen more intentionally. We enable the user with even more control over the generation of their musical piece by training the MusicGen model against a reward model optimized for certain aspects such as danceability, valence, etc. We accomplish this by using a Spotify Research dataset to train MusicGen with Direct Preference Optimization. We evaluate the success of our work through a set of tests that will determine the variety of the music generated, the compliance of creating music according to the specified parameter configuration given, and the accuracy of our customized models.

Introduction

Deep learning models are the center of focus for many applications thought only possible by humans. Apart from many pragmatic applications such as semantic analysis of text or code generation, there are many other creative applications where deep learning is beginning to excel. Tools like DALL-E (Ramesh et al. 2021) and Sora (Liu et al. 2024) prove that with a simple text prompt, many well-trained models are capable of generating multiple different types of creative art forms. One such model that has excelled in the music scene is Meta's MusicGen (Copet et al. 2023): a transformer-based model that can convert either a text or audio prompt into a song. These tools, whether considered inherently creative or not, enable many people to express themselves in a new media form that may not have been possible for them before.

While receiving a new music composition based on a user's input is intriguing, the only amount of creative control the user has is based on the prompt they input. Other facets of the musical creation are left to the randomness of the model itself. This randomness, while suitable for single-use music generation, becomes unrealistic for someone who wants to be intentionally creative while creating generative music. Our project, Music Generation with Direct Policy Optimization, seeks to address this lack of control in generating music by providing a graphical user interface that communicates with various pre-trained models that have been conditioned by direct policy optimization (DPO) (Rafailov et al. 2023) to augment a specific characteristic of the generated music.

In this paper, we explore the efficacy of our proposition by doing the following:

1. We select music from Spotify and YouTube to give us some realistic pairings of good/contemporary songs that coincide with desirable metrics for composing a song.
2. We use DPO to train our MusicGen model to create songs that are closer to the given metrics. We make a different model for each metric.
3. We create a GUI that allows people to continue to use the same text/audio input, but also with added sliders that will average the new models we have trained to a desired output
4. We use CLAP scoring to compare our models with the original MusicGen base model to determine whether our models generate a piece of music that is closer to the intended prompt than the base model.

Related Works

(Justus 2023) provides a framework that relies on human-in-the-loop (HITL) feedback for reinforcement learning. They found that using human feedback in reinforcement learning strengthened the performance of their model and tailored its accuracy to the preferences of the user. While our solution relies on the human rating of audio, we focus on several generalized metrics in our DPO training as opposed to personal preferences. This makes our solution more acceptable to a broader audience while still maintaining control over specific characteristics of the music.

(Cideron et al. 2024) improves upon MusicLM (Agostinelli et al. 2023) by utilizing reinforcement learning from human feedback (RLHF) (Ouyang et al. 2022) to refine the model and create MusicRL. This model improves upon the manual HITL refinement seen in (Justus 2023) with an auxiliary model trained on human feedback preferences to improve the base model. This automated mechanic allows the base model to learn more rapidly and on a larger corpus of preferences. We incorporate the idea of training on a

larger, precompiled list of human feedback in our model. We differ in using DPO to refine our base model and eliminate the need for a reward model.

(Majumder et al. 2024) deployed DPO to enhance an audio generation model based on human feedback. They found that using DPO greatly increased the quality of audio output from their model in both automatic and manual evaluations. Their method differs from ours in three key respects. Firstly, while their model is engineered to produce a wide array of sound events, such as a baby crying or a lion roaring, our focus is predominantly on music generation. Secondly, they implement a diffusion-based model while our method uses a transformer encoder-decoder architecture. Lastly, we concentrate on well-defined key metrics that facilitate the use of pre-labeled data, accelerating dataset preparation and setting clearer performance expectations. Their results, however, indicate that using DPO is beneficial in audio generation tasks.

Methods

Dataset

We originally chose to use the MusicCaps dataset. This dataset contains 5521 music examples labeled with human-tagged aspect lists and a caption. We intended to get YouTube metrics for each of these songs. Through the YouTube API and the Return YouTube Dislike API, you can the the metrics of likes, dislikes, view count, comment count, and rating. With our initial tests, we were not able to get a reward model trained using these metrics. We hypothesize that this was due to the YouTube metrics being disconnected from the actual audio clips from the videos. This could especially be the case for videos that were unrelated to the music (i.e. a commercial or a guitar tutorial).

With these insights, we switched to the Kaggle "Spotify Tracks Genre" dataset (Kaggle 2023). This dataset provides the relevant metrics of: song name, Spotify track ID, artist name, artist ID, popularity, danceability, energy, speechiness, acousticness, instrumentalness, liveness, valence, and tempo, and track genre. Using the track ID, we downloaded the 30 second Spotify song preview. The dataset has 114k songs in 125 genres. After removing duplicates and stale IDs, we were left with about 62.7k songs. We also collected the genre list for each artist using the artist IDs. Using the song name and artist name in this format: "song - artist", we used the YouTube API to search for the song and also downloaded the YouTube metrics as well to augment our data. Between the chosen Spotify metrics and the YouTube metrics, we choose 13 metrics to train DPO models on. These metrics are: popularity, danceability, energy, speechiness, acousticness, instrumentalness, liveness, valence, and tempo, interactions per 1000s views (i/kV), like/dislike ratio, view count, and rating. Interaction per 1000 views was calculated by dividing the sum of likes, dislikes, and comments by the view count. Like/dislike ratio was calculated by dividing the likes by the sum of likes and dislikes.

To create the chosen/rejected audio pairs for the DPO training, we took each song and found the closest song in the genre space. This was done by appending the track genre to the artist genre for each song. While each song could have multiple matches, we only used the first match. It should be noted this will tend to create chosen/rejected song pairs by the same artist, which may or may not be desirable.

DPO

There are several methods to fine-tune a model. Prior to settling on DPO, we considered using Reinforcement Learning with Human Feedback (RLHF). This would use a separate reward model trained on the dataset to determine how well a song (in this case) would fit a metric. Ultimately, we were drawn to DPO as it removes this second model and instead incorporates the rewards internally. This is achieved by passing in a chosen and rejected pair of songs preferentially choosing the logits of the chosen song. This significantly reduces the training overhead for our system as it reduces the number of models we need in half.

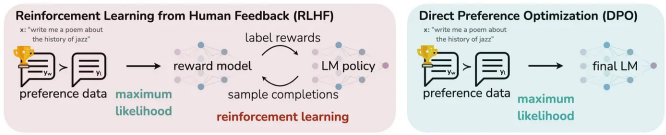


Figure 1: DPO vs RLHF training methods

We chose to use HuggingFace's TRL DPOTrainer code (HuggingFace 2023) to do our DPO trainings. This script is designed for running DPO trainings on text-to-text transformer models. As our MusicGen model is a text/music-to-music model, changes were required to the DPOTrainer code. These changes included a custom tokenizer (meshing the T5 text tokenizer for the text prompts and the tokenized audio), new padded data collator, and extensive shape changes to the DPOTrainer code to handle the four codebooks of the MusicGen model.

To further improve the training process, we added the ability to preprocess and tokenize the audio. The trainer code loads the pre-tokenized audio instead of tokenizing the audio for each audio sample every training. We choose to precompute the reference log probabilities to speed up the training and reduce GPU memory usage. Additionally, we implemented the ability to freeze a variable number of layers in the MusicGen decoder. This allows us to reduce the negative impacts our fine-tuning would have on the model. The MusicGen model already knows what music is and how the prompts relate to that. We only want to tweak the final results. This can be done by freezing more of the front layers.

The DPOTrainer code is compatible with Parameter-Efficient Fine-Tuning (PEFT) (Mangrulkar et al. 2022) and Low-Rank Adaption (LoRA) (Hu et al. 2021). This has been shown to significantly reduce the training time and storage space of models with minimal impact to fine-tuning results. As such, we also chose to use this. There are some other PEFT options that we did not explore. These include DeepSpeed, Quantization, and Gradient Checkpointing.

We trained each of our 13 models as well and the negative models (swapping the chosen and rejected audio) on the `cs`

and `cs2` clusters on the BYU supercomputer. Each job utilized a single GPU, 12 CPU cores, and 120 GB of memory. We trained with a batch size of 3 (note: this is doubled internally as the DPO runs the chosen and rejected audio for each sample), at a learning rate of 1e-8, with 12 frozen layers (out of 24), and with 150 warmup steps. Each model was trained for 10 epochs. This could be sped up significantly as we will later discuss.

Evaluation

For our automatic evaluation, we decided to use CLAP (Wu et al. 2023) to score the outputs of our model based on the song's similarity to two different text prompts. These evaluations both measure important aspects. The first is to ascertain if a song generated by a particular model represents the given metric that that model was trained to encourage or discourage. The second evaluation is meant to measure whether the music continues to match the original text prompt. We want to make sure the music generated still matches the given text prompt.

The dataset used for the evaluation was generated using the following methodology. First, we asked ChatGPT to give us 100 music genres and 3 instruments that fit each genre well. The prompts are given in the following format: genre, instrument1, instrument2, instrument3. This prompting was used as our training dataset used only genres as the prompting and we didn't want to diverge too much from that prompting. ChatGPT decided to go above and beyond what was requested and gave us 115 of these prompts. We then prompted each of our 26 fine-tuned models with these 115 prompts. We set the random seed for each generation to the same value to have a more fair comparison. In preliminary tests, we found that with the random seed set, it is easier to ascertain whether the given model exemplifies the desired metric. We also generated samples from the base model with the same prompts and random seed to act as our base to compare against.

The first evaluation method is to create a text prompt that represents each of our 13 metrics. This is necessary as not all of the metrics correlate directly with the desired behavior e.g. the valence metric is a measure of how high of valence a song has. See Table 1 for a list of the metrics and their associated metric prompt. The CLAP score is computed between each sample generated by our 27 models (including the base model) and these 13 metric prompts.

The second evaluation method is to compare the sampled music to the original text prompt. (Copet et al. 2023) similarly used CLAP scores to evaluate how well the generated samples from MusicGen fit the text prompt. This evaluation is critical as we do not want the models to learn to generate completely different music than the given prompt in favor of maximizing the desired metric.

Graphical User Interface

To develop a tool that enables more creativity while interacting with a generative model, we develop a graphical user interface (GUI) that allows users to fine-tune more aspects of their prompt. The GUI for our project is a simple Flask server with a Bootstrap front end. This makes it easily portable while still supporting a feature-rich widget set with which users can interact. Traditional generative AI interfaces utilize text boxes and written prompts as the sole way to express creativity. Apart from only a text box as seen in Figure 2a, we provide sliders (2b) for specific metrics to provide more fine-grained control than a cleverly crafted prompt. These sliders handle the metrics listed in Table 1. Our Flask server takes the prompt and the values of each of the sliders and creates a file that is put in a specific directory. This directory is a mount point to a remote server with considerable computing power. A script on this server receives and services the request by generating the new song and also the same song with the base MusicGen model. Both songs are generated with the same random seed to make them as similar as possible. Both songs are then delivered to the same shared mount point where the Flask server presents them to the user as seen in Figure 2c.

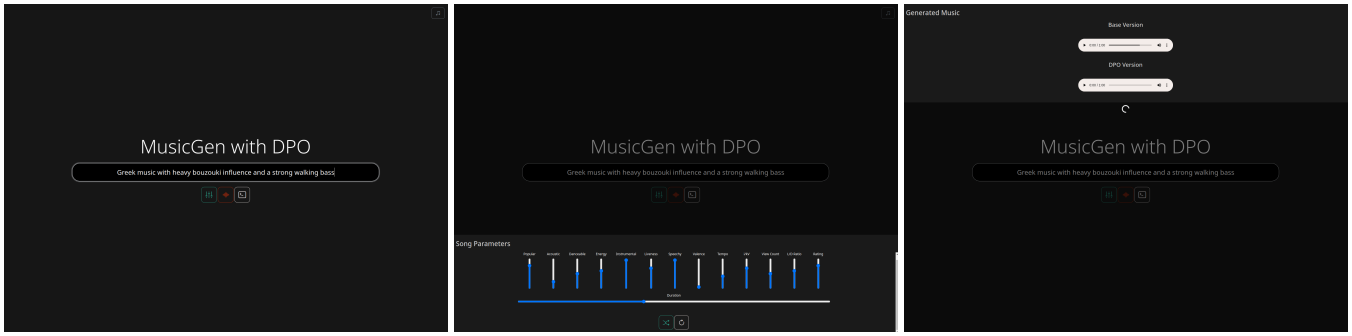
Metric	Metric Prompt
Popularity	popular song
Acousticness	acoustic song
Danceability	danceable song
Energy	energetic song
Instrumentalness	instrumental song
Liveness	song with a live audience
Speechiness	speechy song
Valence	high valence song
Tempo	fast tempo song
Interaction per Thousand Views	interactive song
Like/dislike ratio	likeable song
View count	viewable song
Rating	highly rated song

Table 1: Metrics and their associated metric prompt to be used in the CLAP evaluation.

Model Averaging The sliders in the GUI represent each of the 13 metrics. The slider values range from -5 to 5. To enable this level of control we implement weighted model averaging of the models trained for each metric. If the slider value is positive, we average in the model trained to encourage that metric. Similarly, if the slider value is negative, we average in the model trained to discourage that metric or the negative model. The slider values are then normalized by the sum of the absolute values of all slider values. To ensure consistent weighting when only one or a few sliders is set, we average in the base model with a slider value of 5. This ensures that when a single slider is set to a small value, we don't just use that model exclusively. The models' weights are then combined using the normalized weights. This model is then used to generate the desired music.

Results

Our results seemed to be very mixed. Subjectively some metrics sound correct while other seem to have little to no effect. See Table 2 for sample audio clips. This is not true for all prompts however, some prompts seem to negatively affect the same metrics that sound good. This can be some-



(a) Music generation is initiated with a text prompt.

(b) Sliders control individual metrics that influence the song.

(c) Both the baseline song and the song influenced by the sliders are returned to the user.

Figure 2: Music creation process with the GUI allows for extra control over song characteristics not specified in the prompt.

what explained by examining the Tensorboard results from the trainings as seen in Figure 3. Note that most accuracies are clustered around 0.5 in both training and evaluation. Only a few models exceeded 0.55. This could imply that no learning is occurring. Margins, or the difference between chosen and rejected, are similarly impacted with some margins ending in the negative implying the model is learning the opposite of what it is supposed to.

Metric	Normal	Negative
Base Link		
Energy	Link	Link
Interaction per Thousand Views	Link	Link
Liveness	Link	Link
Acousticness	Link	Link
Like/Dislike Ratio	Link	Link
Popularity	Link	Link
Instrumentalness	Link	Link
Rating	Link	Link
Danceability	Link	Link
Valence	Link	Link
Tempo	Link	Link
View Count	Link	Link
Speechiness	Link	Link

Table 2: Links to audio examples with the prompt "Classic Rock, Electric guitar, keyboard, drums"

We think we can attribute these to primarily two categories: 1) dataset limitations and 2) metric quality. For the dataset limitations, we had to severely downsize our dataset to be able to complete our training. This resulted in some genres being completely removed from the dataset. This could explain why some prompts and genres do not perform as well as others. This is exasperated by the fact that not all metrics mean the same thing in every genre. A popular LoFi song will not look the same as a popular heavy metal song. This brings us to the second category. Not all metrics are good metrics. What makes a highly-rated song? Or what makes a song with a high view count? Is that even related to the song at all? We argue this is not the case. External examples like pop culture or artist name/fame can influence metrics, for example, a Taylor Swift song is going to get more views than an indie pop singer. Metrics that are highly impacted by external factors are going to be much more difficult or impossible to predict.

Evaluation

Our evaluation results, as described previously, were computed using CLAP to measure how well the models enhanced or reduced the metric they were trained to enhance or reduce and how well the sampled music fit the prompt given to the model.

Following the Metric

The purpose of the first evaluation was to determine if our models we able to correctly amplify upon the metrics that they were trained to amplify or in the case of the negative models the metrics they were meant to diminish. Table 3 shows the CLAP scores for each model compared to the base MusicGen model. Ideally, the negative model would be lower than the base, and the Normal model would be higher than the base. Interestingly, both Normal and Negative models tend to be higher than the Base. The Negative models are just as frequently the highest-scoring models compared to the Normal models. These results were very surprising to us as the Negative models were trained to do the exact opposite of the metrics given.

Our theory as to why this occurs is that the model was being trained on music that was in general more of each metric than that of the original MusicGen training dataset. For example, we hypothesize that in general, our dataset was more danceable than the original MusicGen training dataset. This would result in the Negative models achieving higher CLAP scores than the Base. As to why the Negative models outperform the Normal models, we hypothesize that these metrics are not well represented in the dataset. Our method of song pair selection results in most pairs being two songs from the same artist. This can make the metric very hard to learn for the model. As can be seen in the accuracies and margins plots in Figure 3, most of the models didn't learn that much at all.

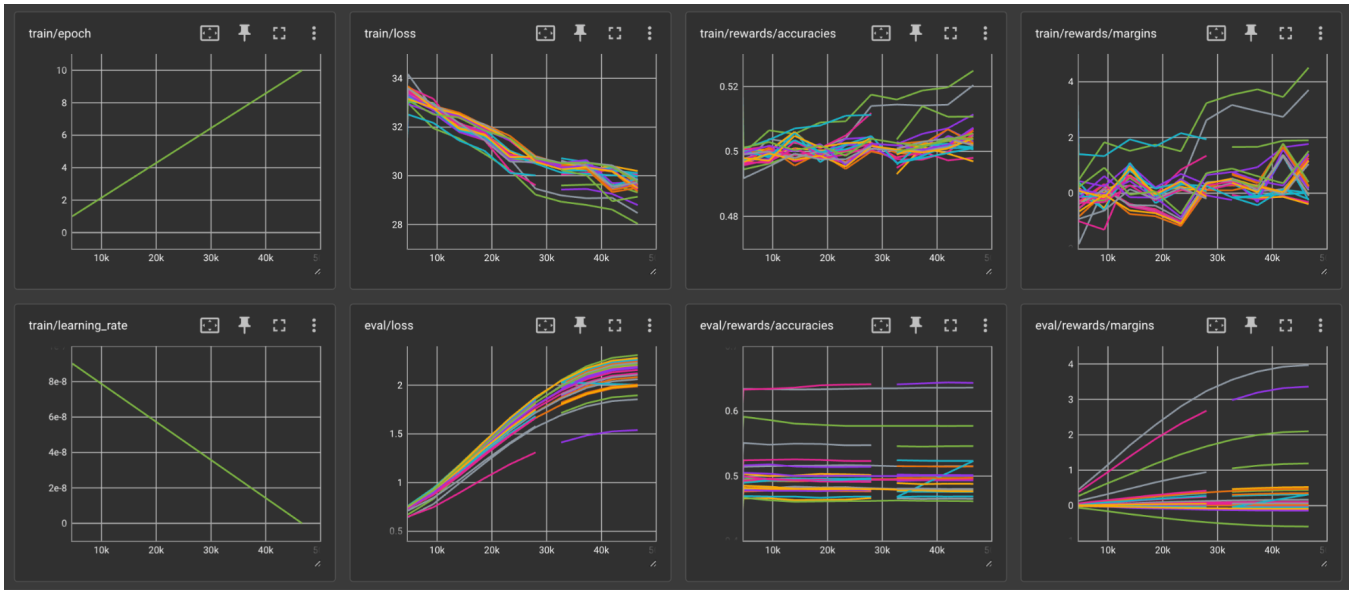


Figure 3: Tensorboard results of DPO Trainings.

To provide a comprehensive evaluation, we also compared all models to the given metric prompts. A heat map for the normal models can be seen in Figure 4 and Figure 5 for the negative models. Unfortunately, there is no discernible pattern. This is likely because many of these metrics are subjective and coincide with one another.

We can see in Table 3 that in 10 out of 13 metrics, our models outperform the base model. This indicates to us that fine-tuning MusicGen with DPO methods can improve the quality of the music.

Metric	Base	Normal	Negative
Energy	**2.32**	2.12	2.25
i/kV	18.44	17.85	**18.56**
Liveness	10.0	10.3	**10.9**
Acousticness	4.62	**5.27**	4.52
LD Ratio	6.51	**6.52**	5.46
Popularity	1.46	**1.66**	1.52
Instrumentalness	19.24	**23.41**	20.86
Rating	**2.12**	1.81	1.84
Danceability	1.07	**1.18**	0.96
Valence	18.32	16.51	**18.71**
Tempo	13.95	14.77	**16.82**
View Count	**7.90**	7.11	7.44
Speechiness	2.61	3.02	**3.31**

Table 3: CLAP score on closeness to metric prompt as seen in Table 1.

Following the Prompt

The purpose of the evaluation is to determine if the DPO training methodology employed in this work results in the model generating music that more closely fits the given text prompt when compared to the base model.

Figure 4 lists the CLAP scores of the normal models compared to the base model. These CLAP scores indicate whether the music generated was consistent with the given prompt. Figure 5 lists the CLAP scores of the negative models compared to the base model. These figures indicate that our models typically match the prompt better than the base model. This can be attributed to the DPO methodology of improving the model on high-quality music, even in the negative case.

Metric	CLAP Score
Speechiness	0.45
Acousticness	0.58
Energy	0.65
Base	0.64
Valence	0.63
Tempo	0.61
i/kV	0.68
Instrumentalness	0.67
Rating	0.73
Popularity	0.72
Liveness	0.74
View Count	0.75
Danceability	0.81
LD Ratio	0.93

Table 4: CLAP score on closeness to the prompt for the normal models. Sorted in descending order.

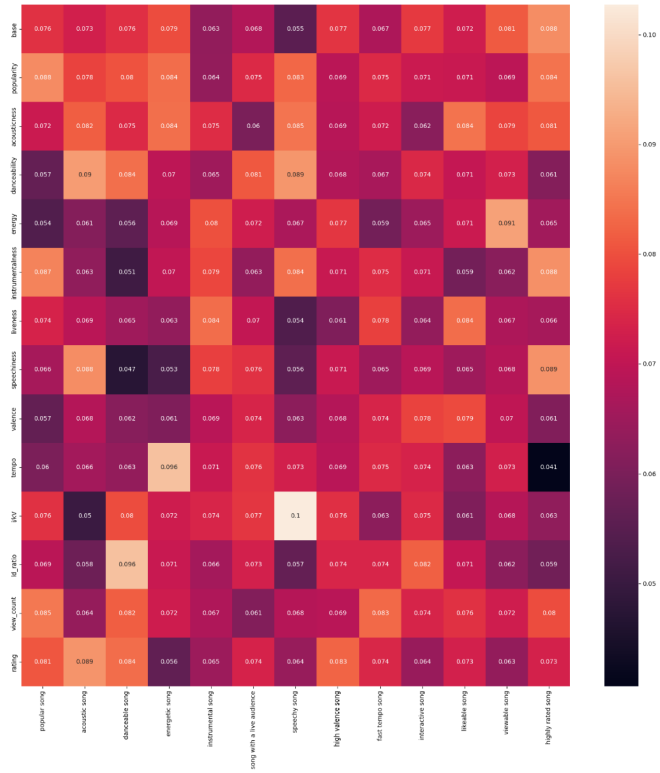


Figure 4: A heat map visualization of the music from each of the models compared to the metric prompts (see Table 1) using the CLAP scores. Each column is normalized to sum to 1.

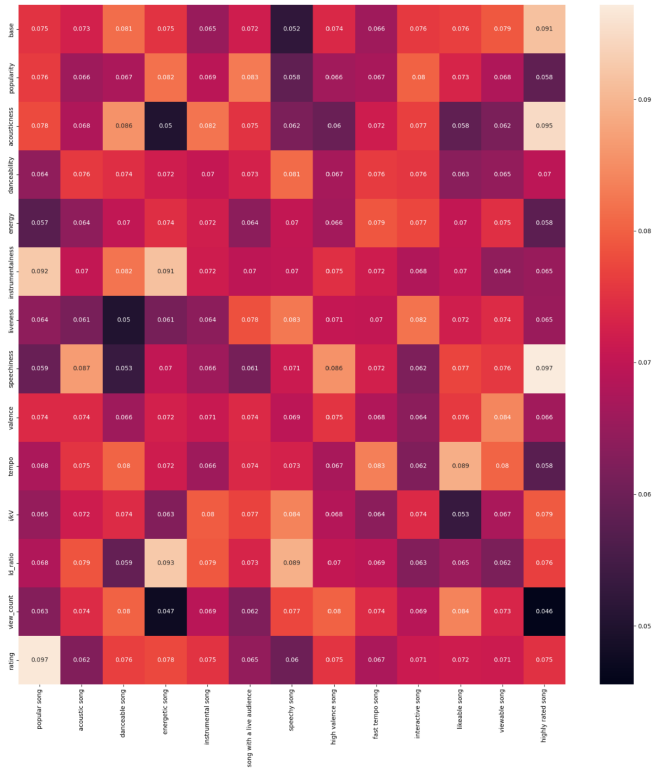


Figure 5: A heat map visualization of the music from each of the Negative models compared to the metric prompts (see Table 1) using the CLAP scores. Each column is normalized to sum to 1.

Future Works

Model

Several aspects can be further explored in training the model. Probably the largest is to properly run the DPO training we needed more computing power. To get our final results we had to significantly decrease the size of the dataset. Originally we had 250k training pairs with 5 unique pairs for each song. This was unable to train even a single epoch within our 24-hour wall time. While reducing the dataset to just a single song per epoch improved this, we ultimately had to restrict the dataset to genres and artist subgenres with the word ”pop”. This reduced the training pairs to 12.5k. If we had more compute we could train on the larger dataset which could potentially produce better results across a larger set of genres.

Another thing compute could improve is in this paper, we focused on fine-tuning the MusicGen small model, however, the MusicGen medium or large produces significantly better music. While training these larger models was infeasible with our available computing power, they may produce better results when trained with DPO.

We ran into many issues in the process of training, some of these issues were not discovered until our training was complete. One such issue was we accidentally left `torch.autograd.set_detect_anomaly()` enabled. Removing this code gave us over a 30% speedup. Before removing this line, adding additional GPUs did not increase our speedup. With this removed, we could increase from 1 to 4 GPUs which would further half our training time. This would allow us to train more epochs or complete more training in a given time (completing each training in about 6 hours instead of 18 on `cs2`). We also discovered that we were requesting far too much memory. This was causing SLURM to limit our jobs to only a few nodes. By reducing our requested memory from 1 TB to 120 GB, we could run far more jobs simultaneously. While fixing the bugs we discovered did increase our computing power and access to nodes, this project could benefit from massive levels of computing power such as the ones available to Meta.

Furthermore, we took far too long modifying the DPO script and getting everything running properly on the supercomputer. While this time was necessary and not wasted, ultimately we ran out of time to explore many options. We significantly underestimated the scope of the project and the effort required to get DPO working. This left us with very little time to run training on the supercomputer. We did some preliminary exploration of the learning rate and number of frozen layers, but this as well as several other meta parameters could be explored in much greater depth.

Negative Metric	CLAP Score
Tempo	0.51
Liveness	0.64
Base	0.64
View Count	0.65
Valence	0.75
LD Ratio	0.81
Acousticness	0.70
i/kV	0.94
Rating	0.94
Danceability	0.91
Instrumentalness	0.95
Popularity	0.98
Speechiness	0.90
Energy	1.04

Table 5: CLAP score on closeness to the prompt for the negatively trained models (e.g. energy model was trained to reduce energy). Sorted in descending order.

Evaluation

The automatic evaluations we performed were informative about how well our model was doing compared to the base model. They are however inconclusive when it comes to whether each model successfully improved in the desired metric or negative metric. To more fully evaluate this, human evaluations are needed. Due to timing constraints, we were unable to perform the necessary human evaluation. This would be a very crucial evaluation as it would greatly improve the validity of our methods.

Another evaluation that we were unable to perform due to timing constraints was a variance evaluation. We hypothesize that fine-tuning the MusicGen model using DPO will also result in a more varied output from the model. This could be measured using CLAP embeddings of songs output by the various models to determine if their output varies more than that of the base model. This evaluation would further verify our argument that our methods improve the creativity of the MusicGen model.

GUI

Our preliminary GUI proved to be efficient in offering precise control over various attributes of the generated music. However, the metrics utilized for refining MusicGen were derived from our Spotify/YouTube dataset, indicating a potential limitation. Consulting a study conducted by professionals in the music composition domain could provide insight into identifying the most relevant metrics to manipulate during music creation. Additionally, exploring research by experts in Human-Computer Interaction to determine the most effective user interface widgets in facilitating the creative process could offer valuable enhancements beyond the current reliance on sliders. Considering these aspects could lead to further refinement and optimization of the user interface, enhancing the overall user experience and creative potential of the system.

Conclusion

In this paper, we propose a system aimed at providing users with more control over the music generation process through the refinement of Meta's MusicGen model. By leveraging Direct Policy Optimization (DPO) and training the model against specific metrics derived from Spotify and YouTube datasets, we sought to enable users to tailor generated music compositions according to desired characteristics such as danceability, valence, and more. We developed a graphical user interface (GUI) to facilitate user interaction, offering intuitive sliders for fine-tuning various musical attributes.

Our methodology involved extensive training of MusicGen models using DPO, followed by evaluation using CLAP scoring to assess the fidelity of generated music compositions. Despite encountering challenges such as dataset limitations and metric quality, our results demonstrated promising improvements over the baseline MusicGen model across several metrics. Notably, our models exhibited enhanced adherence to specified metrics and text prompts, indicating the effectiveness of our approach in empowering users with greater creative control.

While our findings provide valuable insights into the potential of DPO-based refinement in music generation, there remain avenues for further exploration and improvement. Future work could focus on refining training methodologies, expanding dataset sizes, and conducting human evaluations to validate the efficacy of the system from a user perspective. Additionally, enhancements to the graphical user interface and exploration of additional metrics could further enhance the user experience and creative potential of the system. Through continued refinement and innovation, our system has the potential to unlock new possibilities in AI-driven music composition, empowering users to express themselves creatively through generated music compositions.

References

[Agostinelli et al. 2023] Agostinelli, A.; Denk, T. I.; Borsos, Z.; Engel, J.; Verzetti, M.; Caillon, A.; Huang, Q.; Jansen, A.; Roberts, A.; Tagliasacchi, M.; Sharifi, M.; Zeghidour, N.; and Frank, C. 2023. Musiclm: Generating music from text.

[Cideron et al. 2024] Cideron, G.; Girgin, S.; Verzetti, M.; Vincent, D.; Kastelic, M.; Borsos, Z.; McWilliams, B.; Ungureanu, V.; Bachem, O.; Pietquin, O.; Geist, M.; Hussenot, L.; Zeghidour, N.; and Agostinelli, A. 2024. Musicrl: Aligning music generation to human preferences.

[Copet et al. 2023] Copet, J.; Kreuk, F.; Gat, I.; Remez, T.; Kant, D.; Synnaeve, G.; Adi, Y.; and Défossez, A. 2023. Simple and controllable music generation. *arXiv preprint arXiv:2306.05284*.

[Hu et al. 2021] Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2021. Lora: Low-rank adaptation of large language models.

[HuggingFace 2023] HuggingFace. 2023. DPO.py: A script for using Deep Proximity Ordering (DPO) for preferential learning of logits. `https:`

//github.com/huggingface/trl/blob/main/examples/scripts/dpo.py. Accessed: April 24, 2024.

[Justus 2023] Justus, A. A. 2023. Music generation using human-in-the-loop reinforcement learning. In *2023 IEEE International Conference on Big Data (BigData)*, 4479–4484.

[Kaggle 2023] Kaggle. 2023. Spotify tracks genre.

[Liu et al. 2024] Liu, Y.; Zhang, K.; Li, Y.; Yan, Z.; Gao, C.; Chen, R.; Yuan, Z.; Huang, Y.; Sun, H.; Gao, J.; He, L.; and Sun, L. 2024. Sora: A review on background, technology, limitations, and opportunities of large vision models.

[Majumder et al. 2024] Majumder, N.; Hung, C.-Y.; Ghosal, D.; Hsu, W.-N.; Mihalcea, R.; and Poria, S. 2024. Tango 2: Aligning diffusion-based text-to-audio generations through direct preference optimization. *arXiv preprint arXiv:2404.09956*.

[Mangrulkar et al. 2022] Mangrulkar, S.; Gugger, S.; Debut, L.; Belkada, Y.; Paul, S.; and Bossan, B. 2022. Peft: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft.

[Ouyang et al. 2022] Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C. L.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; Schulman, J.; Hilton, J.; Kelton, F.; Miller, L.; Simens, M.; Askell, A.; Welinder, P.; Christiano, P.; Leike, J.; and Lowe, R. 2022. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*.

[Rafailov et al. 2023] Rafailov, R.; Sharma, A.; Mitchell, E.; Ermon, S.; Manning, C. D.; and Finn, C. 2023. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*.

[Ramesh et al. 2021] Ramesh, A.; Pavlov, M.; Goh, G.; Gray, S.; Voss, C.; Radford, A.; Chen, M.; and Sutskever, I. 2021. Zero-shot text-to-image generation. In *International conference on machine learning*, 8821–8831. Pmlr.

[Wu et al. 2023] Wu, Y.; Chen, K.; Zhang, T.; Hui, Y.; Berg-Kirkpatrick, T.; and Dubnov, S. 2023. Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1–5. IEEE.