Efficiently Fine-Tuning MusicGen For Text Conditioned Armenian Music Generation

Author: Hrayr Muradyan

BS in Data Science
American University of Armenia

Supervisor: Karlos Muradyan

Masters in Data Science
University of British Columbia

Composing music is traditionally a realm of human creativity, intuition, and emotion. However, recent advancements in deep learning algorithms have opened directions for the exploration of automated music generation. The primary objective of this research is to create a generative AI model capable of producing moderate sounding compositions that resonate with the cultural nuances of Armenia. We seek to expand the realm of creative possibilities in Armenian music composition and to contribute to the preservation and promotion of Armenian cultural heritage. We employ the MusicGen model for text-to-music generation with efficient implementation of the training pipeline, significantly enhancing both speed and memory utilization. The MusicGen small checkpoint is fine-tuned on a musical corpus of 62.8 hours of Armenian music extracted from the internet. The results were satisfactory where the human evaluators provided on average a rating of 3.84 points from 5 regarding the generated music quality and 3.96 points from 5 concerning the relevance of the given text condition to the generation, indicating the model's ability to generate adequate sounding Armenian compositions aligning with the textual descriptions provided to the model. By pioneering the development of an advanced music generation model for Armenian music, we not only showcase the potential of artificial intelligence in creative domains but also foster Armenian, cultural appreciation and innovation.

Keywords—MusicGen, Generative AI, Armenian Music Generation, Transformers, Music.

I. Introduction

A. Background

Music stands as a profound expression of emotion, culture, and innovation. It is a universal language that went through civilizations with countless transformations. From the invention of musical notation in the ancient past to the rise of digital recording and distribution in the modern age, the way people create, listen, and interact with music has progressed significantly. Armenian music, deeply rooted in the country's rich cultural heritage, reflects a blend of indigenous folk traditions coming from the far past. Traditional Armenian music often features emotive melodies played on instruments like duduk, kanon, and zurna, evoking emotions ranging from joy to sorrow. In modern times, Armenian music has diversified, incorporating contemporary genres like pop, rock, and jazz, while still maintaining its distinct identity and remaining a vibrant expression of the nation's culture.

B. Artificial Intelligence and Music Generation

The process of creating music, which is traditionally an art form deeply rooted in human creativity, intuition, and emotion, has emerged in recent years in the field of artificial intelligence. Deep Learning, in this case, requires special attention, which remarkably outstands traditional machine learning algorithms by the ability to automatically discover complex patterns and relationships within data. Over the past few years, the field of deep learning experienced a sharp evolution bringing groundbreaking advancements and various applications in different domains such as voice recognition, translation, and text-to-speech analysis [1]. The rapid development made algorithms reach into unprecedented realms that previously seemed unrealistic to achieve. One such example is a branch of artificial intelligence focused on creating new data or content that never existed before, called Generative AI, which primarily utilizes deep learning algorithms that proved to be particularly prominent in this field. The process of creating music using deep learning, which is known as music generation, appeared first in the late 1980s as one of the very first efforts to use artificial neural networks in the field [1].

C. Domains For Music Generation

The two most common music generation domains include symbolic and waveform music generation. [2] Symbolic music is represented as discrete data; such as sequences of musical notes or events. The format of symbolic music is standardized and can be easily shared and interpreted by musicians. Waveform music generation refers to the process of creating audio waveforms directly, which represents sound in a continuous domain. While symbolic music is used for music production, controlling stage effects in live performances and studio recordings, the waveform format is commonly used in sound design for film, television, and multimedia projects. People most commonly interact with waveform music, thus, waveform generation is preferable for daily or commercial uses.

D. Deep Learning

As mentioned earlier, deep learning models can automatically learn complex relationships in the dataset and uncover patterns that are not directly observable. The motivation lies in using modern neural network architectures to train a model on a huge musical corpus to automatically learn musical styles which can subsequently be used to generate new compositions. An abstract music generation model would generate a music piece given certain conditions. The algorithm then will iteratively adjust its internal parameters to minimize the difference between its prediction and the original music. This optimization process is guided by a loss function, which quantifies the discrepancy between the generated music and the actual composition. Techniques such as transfer learning, where pre-trained models are fine-tuned on new tasks, can further accelerate the adoption of a deep learning model by alleviating the need for large amounts of labeled data. However, it’s not all smooth sailing - the music generation task is full of challenges and limitations.

E. Challenges and Limitations

Firstly, generating music requires modeling long range sequences [3]. Music is inherently sequential with melodies and harmonies unfolding over time. Thus, each note depends on the previous notes in an autoregressive manner and it is crucial to capture the long-form relationship between notes.

Secondly, unlike speech, music requires the operation of full frequency information [4]. Human perception is very sensitive to music structure and things that do not sound right together are easily noticed. [6] According to Nyquist sampling theorem [5], in order to accurately reconstruct a continuous signal, it must be sampled at a rate at least twice the highest frequency present in the signal. This indicates that in order to fully capture the range of frequencies in music recordings, a higher sampling rate is necessary. While 16 kHz sampling rate is considered more than enough for speech recordings, music samples need closer to 44.1 kHz or 48 kHz, to ensure that they can accurately represent the complex harmonies and melodies [3]. The fact that music recordings need higher sampling rates raises another two important complications.

The sampling rate is the number of data points of a continuous signal that are taken per second to represent the original analog signal in a digital format [4]. Thus, a three-minute recording of a 48 kHz sampling rate will have 8.64 million data points. The large amount of information that is to be processed requires significant computational power and memory resources. Additionally, more sophisticated model architectures and training strategies have to be implemented to effectively learn the high-dimensional music data. The second point refers to the availability of high quality data and the potential increase in the size of the audio files. Storing high sampling rate audio files requires more disk space compared to lower sampling rates and collecting particularly high-quality data can create some difficulties due to lack of availability and high cost.

F. Our Contribution

In this work, we fine-tune the recent MusicGen model developed by Jade Copet et al. for Meta AI [3] with several modifications for better efficiency. MusicGen is a state-of-the-art and powerful model for music generation based on a single language model. We use Armenian music of different genres, moods, and instruments for fine-tuning the model. Our task is to achieve a generative model that is able to compose moderate-sounding Armenian music controlled by a text, which can expand creative possibilities and promote the rich musical heritage evolution into the future, as well as be the starting point for encouraging researchers to concentrate particularly on Armenian music generation.

II. Literature Review

A. History Of Music Generation

The history of music generation is well described in [1]. According to the article, one of the first known cases, even before computers, was attributed to Mozart. The piece of music was created by concatenating randomly chosen pieces of music that were selected by throwing a dice. Computer-based generation started in the late 1950s using probabilistic models like Markov chains and other filtering methods for achieving desired output.

B. One Of The First Deep Learning Models

Peter Todd was one of the first researchers to make use of neural networks for music generation in 1989 [7]. The paper implemented two different three-layer sequential networks that were supposed to learn certain aspects of musical structure from a chosen dataset of musical examples, enabling it to then generate new pieces based on the acquired knowledge. Both networks were designed to exactly recreate the given set of musical examples suggesting that the network has learned the musical structure. The first method involved using time-windowed generations, sequentially by providing the first window as the input to the second window generation. The second method was closer to recurrent network architecture where the input was divided into memory and a plan guiding the network’s actions by identifying the specific sequence being learned or generated.

C. *Modern Deep Learning Models*

Modern deep learning algorithms for music generation can be separated into three main stages:

1. The first stage of modern music generation algorithms includes implementing recurrent neural networks (RNN, LSTM, GRU) for music generation. For years, recurrent neural networks were state-of-the-art approaches for modeling sequential data [8, 9]. However, they all struggle to retain information over very long sequences.
2. The second stage refers to the deployment of Variational AutoEncoders (VAEs) or Generative Adversarial Networks (GANs) for music generation. MidiNet [10] and MuseGAN [11] are some of the most cited examples of symbolic music generation using GANs. MIDI-Sandwich2 is a mix of RNNs and VAE networks for multi-track symbolic music generation. [12]
3. The third stage is the application of transformers in the music domain. Transformers architecture surpassed previous architectures in various sequential data benchmarks and applications. The usage of transformer-based language models is very popular nowadays in music generation due to its unique attention-based mechanism that learns long-term relationships between the time steps. [3, 13, 14]

III. MusicGen

A. *About MusicGen*

MusicGen is a single language model trained for conditional music generation. [3] It provides three available versions: Small, Medium, and Large. The general architecture is the same for all with several modifications in the number of layers and specific layer dimensions. The results show that MusicGen provides satisfactory results showing outperforming results in various benchmarks. The supremacy of MusicGen is defined by its three main building blocks:

1. The use of the modern audio tokenizer model which allows the model to curtail the long, continuous audio representations into short, discrete representations that enable the usage of transformers. This means that instead of dealing with high-dimensional data, it converts the audio signal into a more compact and structured representation. This approach significantly reduces the computational power for processing audio, while still retaining enough information to capture the essential musical features.
2. Being able to be conditioned both by text and melody. This means that given a textual description that matches an audio, generation aims to reflect the characteristics specified by the text input. In other words, the text guides the generation to produce music that aligns with the content, mood, or style described in the prompt. The original MusicGen also supports melody conditioning, which allows longer generations and adding or removing certain attributes of the music from the original one. The scope of this paper, however, excludes the usage of melody conditioning by concentrating on textual guidance. More information about melody conditioning can be found in the original MusicGen paper [3].
3. The usage of a transformer model, which handles the long-term dependencies in the music structure, ensuring that the generation of each next note is firmly dependent on the previous notes. Due to the attention mechanism [16], transformer models weigh the importance of all tokens in a sequence when generating a representation for each next token. Unlike RNNs, transformers access tokens in all positions simultaneously, enabling the model to capture the global meaning of the sequence.

The beautiful trick behind MusicGen is converting the long, continuous audio representation into a low frame and discrete sequence of tokens, then modeling the resulting sequential data with a decoder transformer model. MusicGen uses an EnCodec audio tokenizer model developed by Meta AI [15], that contains three main components: encoder, quantizer, and decoder. The encoder shifts the audio from continuous domain to discrete representation, the quantizer reduces the frame rate even further, while the decoder reconstructs the low frame discrete representation into the original audio with high fidelity.

B. *EnCodec Audio Compression Model*

The architecture of the EnCodec model consists of an encoder-decoder neural network architecture with its latent space quantized using the residual vector quantization (RVQ) technique. Additionally, it uses a lightweight transformer model over the quantized units which reduces bandwidth even further resulting in a compression of the resulting representation by up to 40% [15]. The whole structure is trained in an end-to-end manner. Figure 1 shows the full architecture containing the details for training.

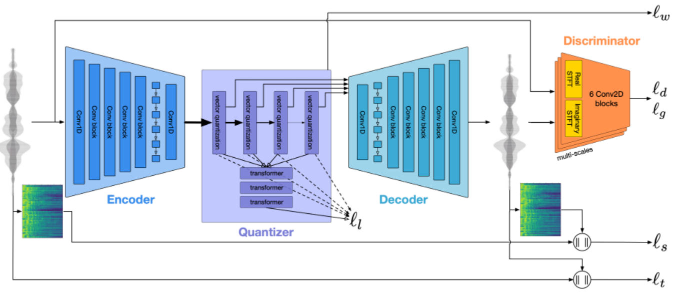

Fig. 1: EnCodec an encoder decoder codec architecture .

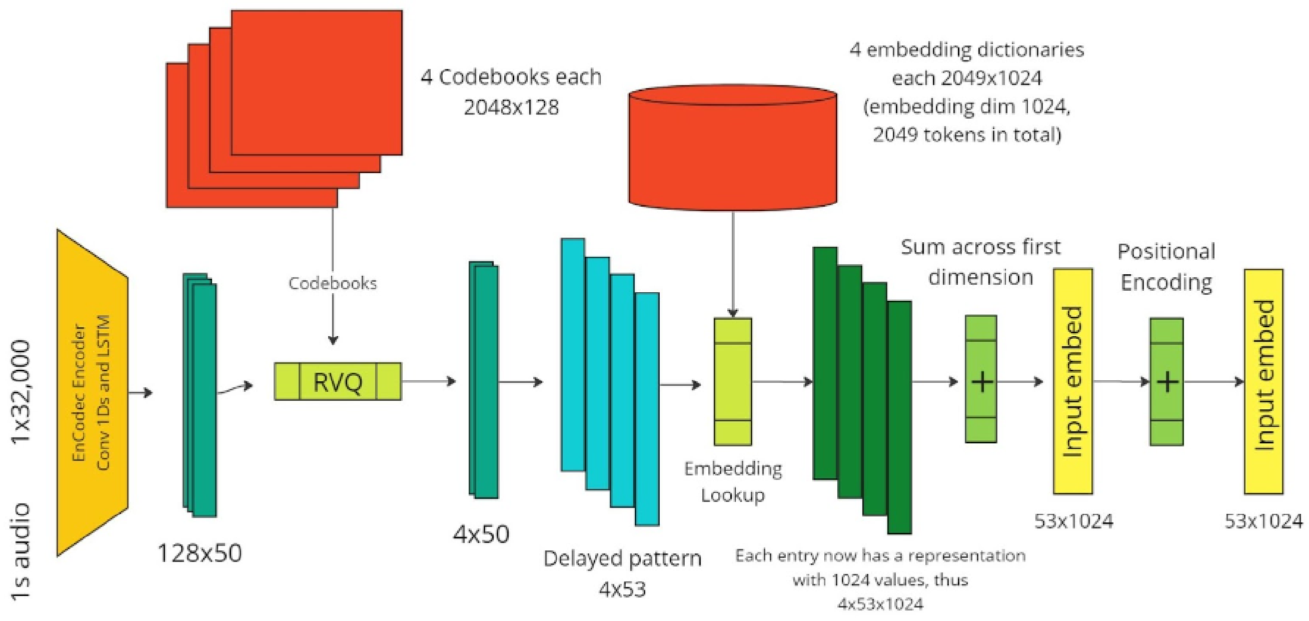
1s audio 1x32,000
EnCodec Encoder Conv 1Ds and LSTM
128x50
4 Codebooks each 2048x128
Codebooks
RVQ
4x50
Delayed pattern 4x53
4 embedding dictionaries each 2049x1024 (embedding dim 1024, 2049 tokens in total)
Embedding Lookup
Each entry now has a representation with 1024 values, thus 4x53x1024
Sum across first dimension
+
Input embed
53x1024
Positional Encoding
+
Input embed
53x1024

Fig. 2: Encoder part of the EnCodec used in the MusicGen.

In the domain of MusicGen, EnCodec is implemented to compress audio data and serve as input for the model during the training procedure. The example operation is shown in fig. 2 with a sampling rate 32,000; a number of codebooks 4; a codebook dimension 1024; a codebook size 2048; and a delayed pattern. The results show that EnCodec has superior performance providing a state-of-the-art approach for audio encoding which was used to convert a 1-second, 32,000 sampling rate audio to 4x50 discrete representation reducing the dimension significantly. More about EnCodec can be found in the original paper [15].

C. *T5 Text Conditioner*

To be able to pass the information from text to the audio generation model, MusicGen suggests either a pre-trained text encoder T5 [17] developed by Google, a slightly improved, instruction-based encoder model called FLAN-T5 [18] which showed a bit better performance than T5 in various metrics, or a joint text-audio representation called CLAP [19].

The original paper experiments with all three options but concludes T5 surpassing the other two, which is then used as the main encoder for all MusicGen models. Specifically, the T5 base model is used for text tokenization, which converts text to a matrix of shape *(n_tokens, embedding_dimension)*. Embedding dimension in the base model is 768, which explains the contextual information of the token in the text. To match the default dimension of the EnCodec and language model, an additional linear layer is applied after the tokenization to shift from 768 to D dimension, where D is 1024, 1536, 2048 for the small, medium and large models respectively. The figure 3 shows an example of the whole procedure.

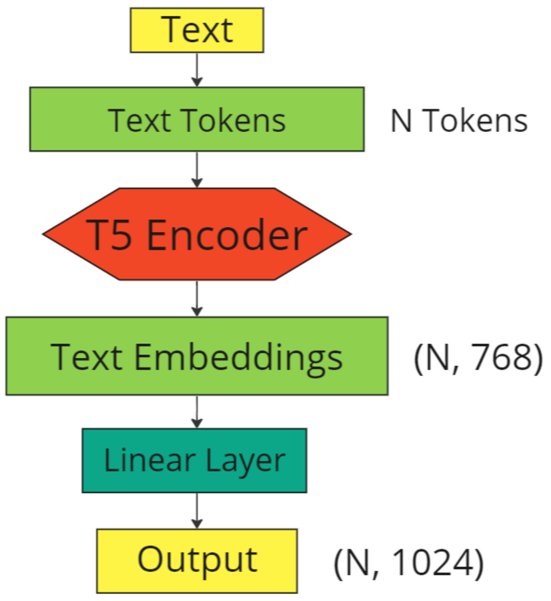

Fig. 3: Text conditioning using T5 Tokenizer and a Linear Layer for the MusicGen small model.

D. *Transformer Model*

As mentioned above, the core model is a decoder-only transformer model. In contexts where the task primarily involves generation or autoregressive generation, the decoder component of the transformers is used. The model contains L layers and dimension D, which both depend on the size of the model. For the small, medium, and large models, L is 24, 48, 48 and D is 1024, 1536, and 2048 respectively. Each layer is composed of a causal self-attention block and several linear layers with layer normalizations. It then uses a cross-attention block that receives input from the conditioning signal C, which stands for the representation acquired from T5.

IV. Our Method

A. General Modifications

For some reason, the number of parameters provided in the paper does not match the actual number of trainable parameters for the models. Table 1 provides the submitted parameters and the actual number of trainable parameters. Such heavy, large-scale models are hard to train due to their computational demands and time requirements. The sheer number of parameters in these models necessitates extensive computational resources.

Category/Model	**Small**	**Medium**	**Large**
Parameters (in the paper)	300 M	1.5 B	3.3 B
Actual Parameters	~400 M	~1.8 B	~3.2 B

Table 1: Parameters provided in the paper vs. actual parameters of the model for LM model only.

Moreover, training such models often involves prolonged periods, ranging from days to weeks, depending on factors such as dataset size, model architecture, and available computing power. Additionally, optimizing hyper-parameters and conducting experiments to fine-tune the model further extend the training time. Such conditions underscore the importance of efficient implementation and infrastructure to accelerate the training process, making it more accessible to researchers and practitioners. Loading the complete model with all three blocks has huge memory requirements (see Table 2).

Model name/Category	**Loading**	**30s. Generation**	**Training**
MusicGen Small	1.69 GB	3.75 GB	OMM
MusicGen Medium	4.5 GB	7.63 GB	OMM
MusicGen Large	7.2 GB	OMM	OMM

Table 2: The memory consumption by each model when loading, 30-second music generation and training on 8 GB GPU.

We simplified the training pipeline to make training more accessible which reduces computational complexity by enabling the system to allocate resources more efficiently. This will further facilitate faster training and optimization of the model. In the original implementation, the dataset expects 30-second music files and their corresponding information in the dictionary saved as a JSON format. In each iteration, music files are read and preprocessed with EnCodec, also, JSON files are read, tokenized, and embedded using T5 Tokenizer (see figures 2 and 3). The training pipeline exhibits redundancy by loading and processing the same text and encoding the same audio data in every training step. The repetitive processing of identical data introduces unnecessary computational overhead and inefficiency into the training process. Thus, we can eliminate the need to repeat these computations in every epoch by precomputing and caching the encoded text and compressed audio representations before training begins (see Figure 4). The optimization not only accelerates the training process but also saves memory by excluding the need to load T5 and EnCodec.

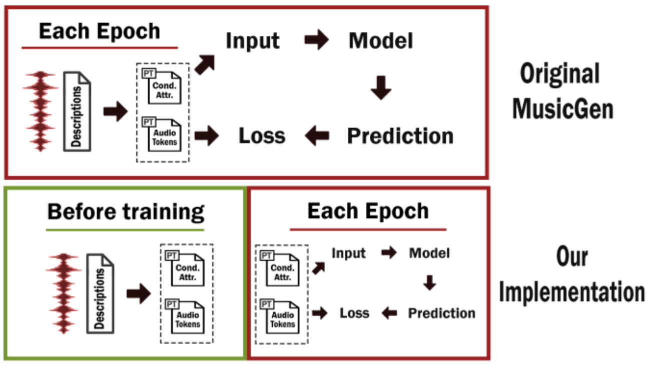

Fig. 4: The training pipeline of the original MusicGen vs. our implementation.

We created a new data class that would expect preprocessed files instead of the raw music files and JSON attributes. The preprocessing step was done before starting the training process and the corresponding preprocessed files were saved in a separate folder. One important difference is that the original implementation randomly chooses 30-second chunks from a randomly selected track for training, while our method extracts fixed 30-second portions from each track.

<u>Note:</u> *It is important to note that the original code applies token-level dropout to the descriptions during the training process before preprocessing the files. In other words, in each iteration, each word has a certain probability of being omitted from the description during the iteration and only after that they are prepared. This technique adds regularization to prevent overfitting. However, our method would make it impossible to implement, since we save already preprocessed files. To have something similar to that, we chose to drop out the embeddings of each token with the same logic. Nevertheless, this option is not the same as the original one since embeddings contain information about the context of the whole sentence, and removing an embedding of a certain word doesn't mean that the information of that word is fully removed from the sentence.*

To illustrate the changes on the small model of MusicGen, we calculated the number of parameters before and after the changes, as well as computed the average training time for an epoch with our method for the small model. From the 420 million parameters, the transformer model itself accounts for roughly 402 million. Additionally, the T5-base model contains 220 million parameters. So, in total of 238 out of 640 million is not loaded with the modification. A 10,000 iteration through the data on the small model with original implementation takes about 110 minutes, while our model decreased that number up to 70 minutes. We additionally added a feature to freeze layers up to the N-th layer in case there is lesser computational power available.

B. Dataset

The data preparation requires special attention. It has been revealed through various experiments that the data collection process requires the most careful approach. Unfortunately, the original documentation does not include examples of descriptions from the training dataset. The only option was to compare the collected descriptions with the data included in the MusicCaps dataset which was used for one of the benchmarks in the paper. It contains 5,521 music examples, each of which is labeled with a free text caption written by musicians. The annotations were very detailed, done by experts, whose knowledge we didn't possess. Additionally, we had limited human resources and could not allocate months for data labeling. To make the process realistic for us, we used an automatic music labeling library called Essentia. Specifically, the checkpoints predicting the mood, genre, and instruments using classification. However, we still couldn't rely solely on its predictions because they were poor in some cases, originating a need to exclude some classes from the list. Another major limitation represents the models' inability to detect Armenian instruments and certain genres because it was not trained on examples containing Armenian compositions. It even did not contain the classes for the most common Armenian instruments. The mood, on the other hand, is mostly universal and is applicable to any type of music. Dialogue with Armenian music expert Artur Avanesov showed that labeling Armenian music genres is a harder task than we thought, besides that, Essentia did not hesitate to label all music samples incorrectly as classical music compositions. Thus, instruments are either provided explicitly and/or predicted by Essentia, genre is provided manually and mood is predicted fully using Essentia. It should be noted that predictions are done not on the full track, but on trimmed 30-second clips. To have more accurate descriptions, we exploited three different types of descriptions:

- **Default_description:** A hard-coded description with the same, default structure where genres, instruments and moods are passed to placeholders.
- **Creative_description:** Genres, instruments, and moods are given as a prompt to the Falcon 1B instruct model that is given an instruction to generate a creative description.
- **Manual_description:** A description given by the user, fully manual, to the music piece.

Shorter descriptions are all padded up and longer ones are shortened down to the length of 55 tokens. All three descriptions are processed and saved in a single file with three dimensions, e.g. (3, 55, 1024). When accessing an element from the dataset class, the file is loaded and a random processed description is taken for training. The usage of three different descriptions for the same output introduces a mix of regularization and data augmentation. Examples of the descriptions used can be found here.

The dataset contains 62.8 hours of raw data extracted from YouTube. To increase the samples in the dataset, the clipping was done with overlapping portions with a 15-second sliding window. The final training dataset consists of 14,637 clips, all 30-seconds long, resulting in roughly 120 hours of music. We also convert the stereo examples to mono. More information about how to extract data and prepare it can be found here.

C. Data Analysis

The magnitude of our data is relatively small compared to the original dataset collected by Meta AI. MusicGen was trained on 20K hours of licensed music with about 10K high-quality music tracks and 390K instrument-only music extracted from Shutterstock and Pond5 [3]. While some generations, for instance, piano or violin music, can be easily learned with small data due to fine-tuning, as the original data contained a bunch of such examples, learning from scratch some specific Armenian instruments like duduk, tar or kamancha would be problematic with little data. Figure 5 shows the most frequent instruments, moods, and genres appearing in the data, suggesting their potential efficacy in achieving the desired results.

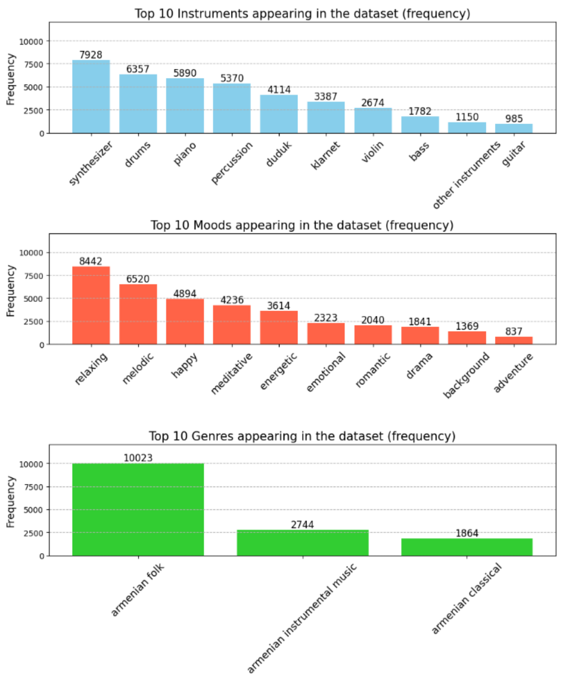

Fig. 5: Most frequent appearing instruments, moods and genres in the dataset.

Additionally, it is important to note that supremacy of a specific label and the overall imbalance in either of the categories can introduce bias for the model due to the small amount of data. Moreover, note that most compositions are not instrument-only, but contain mix of various instruments.

D. *Training*

The term "epoch" in the MusicGen implementation does not refer to a complete iteration over the entire training set. Instead, another parameter called *updates_per_epoch* specifies the number of batch-sized iterations through the training set during each epoch. We trained the MusicGen small model on a Nvidia GeForce RTX 4070 TI Super with 16 GB memory. Every training is very expensive that's why we didn't play around with the hyper-parameters that much and most of them were left as they are. We followed the same training pattern as the MusicGen paper suggested, with the first 10 epochs trained using the AdamW optimizer with a learning rate of 10^{-4}, then using the Dadam [20] optimizer by Meta AI. An additional 6 epochs were trained using the Dadam optimizer. A batch-size of 6 is taken with 10,000 updates per epoch, meaning that each epoch iterated over the whole dataset approximately 4 times.

V. Results

Our model is evaluated using a subjective metric. We questioned human evaluators about the overall quality of the generation and its relevance to the text input. Each rater was provided with the same 4 prompts taken from the validation set, where for each prompt there are three categories:

- Our model generation conditioned on the prompt,
- MusicGen generation conditioned on the prompt,
- The original track corresponding to that prompt.

Thus, each evaluator provided feedback about 12 music samples. For both our model and MusicGen, we took the first immediate generation. Each category was provided randomly, in different orders and in a blind manner, meaning that the evaluators were not aware of the existing groupings. Additionally, they were required to submit a rating of how closely they think the prompt aligns with the generations and the original track. Moreover, we provided prompts indicating different genres, moods, and instruments. The rating scale ranged from 1 to 5 for both metrics, where 5 is the highest positive rating and 1 is the lowest negative one. In total 43 people with various backgrounds and understanding in music took part in the evaluation process providing 1,012 responses. Our model provided rewarding results with about 3.84 quality rating and 3.96 relevance rating. Table 3 shows the results averaged for each respondent and sample.

Model/Test	**Quality**	**Relevance**
MusicGen	3.2	3.173
Our Model	3.845	3.958
Original	4.14	4.158

Table 3: The average of the ratings provided by the evaluators for our model, MusicGen and original composition.

The evaluation of the MusicGen model suggests that it did not have enough examples from Armenian compositions in the training set, which resulted in the absence of proper understanding of Armenian instruments and style. In contrast, the fine-tuned model emerged as a notable improvement over the original MusicGen model. The incorporation of Armenian-style music enabled the model to generate better quality music, being closer to the original compositions standing as the human baseline. It is also worth noting that our model's performance converges to the results provided in the MusicGen paper [3], where the small MusicGen model got on average 3.96 quality rating and 4.05 relevance rating.

VI. Discussion

A. *Further Directions*

The experiment provides much room for discussion. The limitation provided by MusicGen official paper is the absence of detailed control over how closely the output should match the instructions provided. Additionally, it discusses the ethical concerns connected to the dataset acquisition and an unfair competition for artists due to inclusion of generative models. These are, certainly, open problems that need careful attention. Another further research direction that can be a nice feature to have, is to be able to weight training samples highlighting compositions that are particularly admired by a broad audience. In other words, there are compositions that are widely revered by a large number of people, containing melodic beauty, harmonic richness or structural mastery, while some examples have lower quality and fail to attract listeners. By assigning weights to compositions, the model could prioritize learning from popular and esteemed pieces, potentially leading to the generation of music that resonates deeply with audiences, feels genuine, expressive, and emotionally compelling. This approach holds promise for improving the model's ability to create compositions that capture the essence of beloved musical works.

B. *Limitations Of Our Work*

While our music generation model has successfully achieved the aims outlined in the paper, we want to stress several limitations to be addressed in the further experiments. Firstly, the most significant constraint is the labeling of the dataset used for training. While we tried to maximize the quality of the labels with the resources we had, our knowledge for labeling complex musical compositions was too little. Because of that, the labels turned out to be too general, introducing learning biases or overlook nuances that affected the model's learning process. This further degraded the ability of the model to align deeply with the desired creative vision provided by the text. Therefore, addressing issues related to labeling quality is essential for enhancing the text-conditioned control of music generation. Armenian music is complex and encompasses a wide range of styles, genres and especially influences. Better labeling could be achieved by involving multiple experts with sufficient musical knowledge in the process, ensuring well-informed and well-detailed descriptions. Unfortunately, we didn't have those resources. Secondly, despite our best efforts to collect a moderate size dataset, there was little amount of data available on the internet. While we fine-tuned the MusicGen model, some instruments were still new for the model and it had to learn their sound from scratch. Addressing these limitations will be crucial for advancing the capabilities of our music generation model and ensuring its relevance and effectiveness in diverse musical contexts. Additionally, it is critical to mention that while the MusicGen development team had an agreement with Shutterstock for using their data, we had no proper authorization to use YouTube's data. Scraping data from YouTube raises ethical considerations due to potential violations of terms of service and copyright regulations, since some compositions may be protected. Thus, future researchers should prioritize the development of more robust labeling methodologies and the expansion of datasets to encompass a broader spectrum of musical styles, enabling the model to further refine its understanding and creativity in music generation tasks, as well as follow the ethical guidelines to ensure they have the authority to use the data under the hand.

VII. Conclusion

We fine-tuned MusicGen for text-conditioned Armenian music generation that has yielded promising results, aligning with the objectives outlined in this paper. By successfully producing a generational model, we have taken a significant step towards preserving and promoting the rich musical heritage of Armenian music. Additionally, our contribution serves as a foundational baseline in the emerging field of Armenian generative music, as one of the first steps in the domain. Moreover, our study also reveals certain limitations and directions for further research. Enhancing the quality and diversity of the dataset through the allocation of additional resources stands out as a major step towards further improving the efficiency of the Armenian music generation model and contributing to the broader landscape of computational creativity in music composition. The significance of this research extends beyond its technical achievements. Ultimately, our work underscores the potential of artificial intelligence to contribute meaningfully to the preservation and innovation of cultural musical traditions. Lastly, we want to thank the developers of the MusicGen for their amazing contribution and open-source codes.

VIII. References

[1] Briot, J.-P. (2020). From artificial neural networks to deep learning for music generation: history, concepts and trends. Neural Computing and Applications, 33(1), 39–65. doi:10.1007/s00521-020-05399-0

[2] Vinet, H. (2004). The Representation Levels of Music Information. In: Wiil, U.K. (eds) Computer Music Modeling and Retrieval. CMMR 2003. Lecture Notes in Computer Science, vol 2771. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-39900-1_17

[3] Copet, Jade, et al. Simple and Controllable Music Generation. arXiv:2306.05284, arXiv, 29 Jan. 2024. arXiv.org, http://arxiv.org/abs/2306.05284

[4] Müller, M. (2015). Fundamentals of Music Processing. doi:10.1007/978-3-319-21945-5

[5] Colarusso, Pina, et al. "Raman and Infrared Microspectroscopy." Encyclopedia of Spectroscopy and Spectrometry, Elsevier, 1999, pp. 1945–54. DOI.org, https://doi.org/10.1006/rwsp.2000.0402

[6] Fedorenko E, McDermott JH, Norman-Haignere S, Kanwisher N. Sensitivity to musical structure in the human brain. J Neurophysiol. 2012 Dec;108(12):3289-300. doi: 10.1152/jn.00209.2012. Epub 2012 Sep 26. PMID: 23019005; PMCID: PMC3544885

[7] Todd, P. M. (1989). A Connectionist Approach to Algorithmic Composition. Computer Music Journal, 13(4), 27. doi:10.2307/3679551

[8] Mangal, Sanidhya, et al. "LSTM Based Music Generation System." IARJSET, vol. 6, no. 5, May 2019. https://doi.org/10.17148/IARJSET.2019.6508

[9] Conner, Michael, et al. Music Generation Using an LSTM. arXiv:2203.12105, arXiv, 22 Mar. 2022. arXiv.org, http://arxiv.org/abs/2203.12105

[10] Yang, Li-Chia, et al. MidiNet: A Convolutional Generative Adversarial Network for Symbolic-Domain Music Generation. arXiv:1703.10847, arXiv, 18 July 2017. arXiv.org, http://arxiv.org/abs/1703.10847

[11] Dong, H.-W., Hsiao, W.-Y., Yang, L.-C., & Yang, Y.-H. (2018). MuseGAN: Multi-track Sequential Generative Adversarial Networks for Symbolic Music

Generation and Accompaniment. Proceedings of the AAAI Conference on Artificial Intelligence, 32(1). https://doi.org/10.1609/aaai.v32i1.11312

[12] Liang, Xia, et al. MIDI-Sandwich2: RNN-Based Hierarchical Multi-Modal Fusion Generation VAE Networks for Multi-Track Symbolic Music Generation. arXiv:1909.03522, arXiv, 8 Sept. 2019. arXiv.org, http://arxiv.org/abs/1909.03522

[13] Ens, Jeff, and Philippe Pasquier. MMM: Exploring Conditional Multi-Track Music Generation with the Transformer. arXiv:2008.06048, arXiv, 20 Aug. 2020. arXiv.org, http://arxiv.org/abs/2008.06048

[14] Agostinelli, Andrea, et al. MusicLM: Generating Music From Text. arXiv:2301.11325, arXiv, 2023. arXiv.org, http://arxiv.org/abs/2301.11325

[15] Défossez, Alexandre, et al. High Fidelity Neural Audio Compression. arXiv:2210.13438, arXiv, 2022. arXiv.org, http://arxiv.org/abs/2210.13438

[16] Vaswani, Ashish, et al. Attention Is All You Need. arXiv:1706.03762, arXiv, 1 Aug. 2023. arXiv.org, http://arxiv.org/abs/1706.03762

[17] Raffel, Colin, et al. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. arXiv:1910.10683, arXiv, 19 Sept. 2023. arXiv.org, http://arxiv.org/abs/1910.10683

[18] Chung, Hyung Won, et al. Scaling Instruction-Finetuned Language Models. arXiv:2210.11416, arXiv, 6 Dec. 2022.arXiv.org, http://arxiv.org/abs/2210.11416

[19] Wu, Yusong, et al. Large-Scale Contrastive Language-Audio Pretraining with Feature Fusion and Keyword-to-Caption Augmentation. arXiv, 21 Mar. 2024. arXiv.org, http://arxiv.org/abs/2211.06687

[20] Defazio, Aaron, and Konstantin Mishchenko. Learning-Rate-Free Learning by D-Adaptation. arXiv:2301.07733, arXiv, 7 July 2023. arXiv.org, http://arxiv.org/abs/2301.07733.