A Transformer Based Playlist Creation System Through Genre Classification

Colin Houde
CAP 6711 – Spring 2024
University of Central Florida

1. Introduction

The modern transformer architecture within machine learning models has paved the way for all kinds of systems to be developed. With its unprecedented way to learn important features about all kinds of data, the transformer allows for a completely new approach to solving new and existing problems. The most popular widespread version of the transformer architecture found today is very prominently the Large Language Model (LLM). Spreading across many variations, the LLM has taken the world by storm and has yet to touch the surface in its true potential across all industries.

While transformers are often celebrated for their capabilities in natural language processing, their versatility extends far beyond text-based applications. The core strength of the transformer lies in its ability to model complex, long-term dependencies within sequential data. This makes it an excellent candidate for tasks involving structured data such as audio signals, where identifying patterns over time is crucial.

In the realm of audio signals, audio files can be represented as spectrograms—visual representations of audio frequencies over time. This transformation of raw sound data into a structured, image-like format enables the ability to apply transformer-based models. By capturing temporal and spectral relationships in audio, some models can effectively learn to distinguish between different genres of music, including nuanced subgenres. Which can further be used to produce high accuracy classification and high-quality playlists centered around a given input song.

In this work, I explore the implementation of an Audio Spectrogram Transformer (AST) model for a music playlist generation system. This is done by leveraging the power of the transformer architecture by fine tuning the model to identify music genres and produce high quality embeddings ultimately for a playlist recommendation system designed around Electronic Dance Music (EDM.

2. Related Work

The first related work [1] pertains to the original ‘AST: Audio Spectrogram Transformer’ paper. The goes into detail on the architecture, the design of the AST model and why they chose to part from the more common CNN architecture when working with audio. This paper also discusses various experiments used when originally testing the model. The experiments show how the model preforms well in audio classification and how the transformer architecture is beneficial when deploying to tasks involving audio. This paper was the reason I chose to use this model and helped me better understand the model itself.

More pieces of work have stemmed from the original AST paper and have evolved the original idea. Variants of the AST have been deployed to tackle other ideas such as self-supervised learning with masked spectrogram reconstruction [2]. This variant of the model is designed to predict the missing portion of the masked spectrogram which in turn better understands the dataset being trained on.

Other groups have tackled the same music classification task using non-transformer-based architectures such as CNN models [3]. Using CNN’s this group was able to achieve similar results as I have using 3-second audio sampled spectrograms. While part of it is essentially the same idea, they did not use a transformer-based method and ultimately did not use the system for a playlist creation system.

Another interesting piece of work presents an NLP approach to music classification and recommendation [4]. While not using AST or spectrograms, this paper describes a way to recommend music based on the lyrics of the song in question. While not exactly the same approach, they too also use a transformer-based BERT model to achieve a similar goal.

3. Method

The methodology behind this implementation stems from wanting to produce a music playlist creation

system that is designed for DJ's and people who enjoy listening to Electronic Dance Music (EDM). Many popular playlist creation systems now adays use a mix of algorithmic methods, human curation metadata matching and user data collaborative filtering to create playlists around a given input song. This works great for music that has ample amounts of meta data and systems that have large amounts of user data. But it does not work well for music that lacks meta data and for new, less popular songs.

Electronic Dance Music inherently on average has less meta data than other popular genres. Meta data in regard to music can be a multitude of things ranging from the genre labels associated with the song, the lyrics of the song and the era the song came from.

Instrumental tracks (songs without lyrics) are common in EDM due to its origins as dance-focused music, where the emphasis is often on rhythm and melody rather than lyrics. Subgenres like trance, techno, and house music frequently feature instrumental compositions without any lyrics at all. This makes it especially difficult for popular playlist creation systems to produce EDM playlists of similar music as they don't have as much meta data to cross reference with.

This is where my system comes in. A system that does not use any meta data or user data and only recommends songs based on the similarity of the embeddings after a forward pass through the transformer-based model that I have fine-tuned. This fixes a few things in relation to creating playlists within the EDM genre. It allows for a much simpler pipeline to create playlists as well as better recommendations for music that does not have any lyrics or that simply lacks metadata. It also helps recommend music that is not quite as popular as the most played songs – keeping your playlists fresh and new.

3.1 The Pipeline

The pipeline of this playlist creation system is quite simple and can be applied to any range of music genres the end user would like. The implementation of this project is designed to seamlessly integrate the capabilities of the Audio Spectrogram Transformer (AST) with the task of personalized playlist creation by fine-tuning for genre classification. The pipeline proceeds through the following stages:

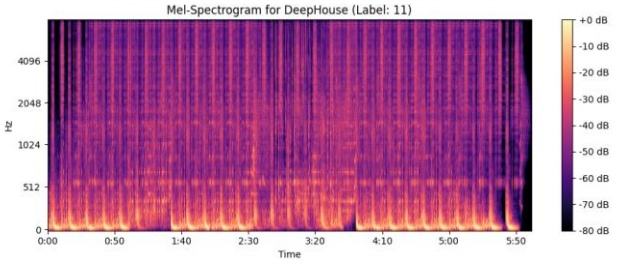
1. **Defining Target Subgenres**: The process begins by identifying the subgenres the end user is most interested in. These chosen subgenres form the foundation for the playlists to be generated.
2. **Data Collection and Preparation**: To fine-tune the AST model, a dataset comprising of songs from the selected subgenres is gathered for the model to train on and later classify test songs accurately.
3. **Fine-Tuning the AST Model**: The AST model, pre-trained on general audio tasks, is fine-tuned on the curated dataset for classification. This enables it to learn nuances between the specified subgenres and to produce more specialized embeddings to later compare.
4. **Embedding Generation for the Music Database**: Once fine-tuned, the AST model is deployed to produce embeddings for all songs in a central database. This database serves as the repository from which the end playlists are built from.
5. **Similarity Computation and Playlist Generation**: Using cosine similarity, the embeddings of the query song are compared against all embeddings in the database. The system identifies the k-most similar tracks based on this metric and returns them as the user-requested playlist, ensuring all songs feel and sound the same – a playlist of similar songs!

3.2 Dataset Curation

For this system to work effectively, it is important to curate a dataset using the genres of music the end user wants to create a playlist around. In a perfect world, where all music is free and everyone had unlimited storage and compute power, it would be best to train the model using all music across all genres. But for my sake of limited storage and access, I limited my dataset to just the genres I knew I wanted to create playlists around. Specifically, the genres I chose to fine-tune the model on were the following: Afro House, Bass House, Deep House, Deep Tech, Downtempo, Drum and Bass, Funky House, House, Melodic house, Progressive House, Tech House, Techno, Techno-Raw and Trance. All of these genres are subgenres under the 'Electronic Dance Music' umbrella. I chose to do it this way because I knew these are the subgenres I like to listen to and fine tuning on these would produce a system that I would actually use right now and in the future on a practical bases.

3.3 Spectrograms

Audio data, in its raw waveform form, presents significant challenges for machine learning models due to its high dimensionality and lack of explicit structure. To address this, raw audio is first transformed into a spectrogram - a visual representation that maps sound frequency to time.



Spectrograms are created using the Short-Time Fourier Transform (STFT), which breaks the audio signal into overlapping windows and computes the frequency components within each window. The result is a 2D representation, where one axis corresponds to time, the other to frequency, and the intensity of each point represents the amplitude of a specific frequency at a given time. Although not the audio file itself, these spectrograms are a derivative of the audio file and are a high-quality representation of the given audio. These spectrograms are then used as input for the AST model.

3.4 Audio Spectrogram Transformer

The Audio Spectrogram Transformer (AST) is a deep learning model designed to process audio data in the form of spectrograms. It builds upon the Vision Transformer (ViT) architecture, adapting its patch-based processing and self-attention mechanisms to analyze the time-frequency representations of sound.

The AST model processes the spectrogram as input, similar to how a ViT model processes images:

1. **Patch Partitioning**: The spectrogram is divided into P×P patches, with each patch flattened into a 1D vector.
2. **Linear Projection**: Each patch is passed through a linear embedding layer, projecting it into a fixed-dimensional feature space.
3. **Positional Encoding**: Positional encodings are added to each patch embedding to retain information about the spatial arrangement of patches in the spectrogram.

After these patches are fed through the transformer encoder, the processed tokens are passed through a classification head and embeddings are taken to compute similarity.

3.5 Fine-Tuning

Fine-tuning the AST model for genre classification is a crucial step in achieving the nuanced understanding necessary for an effective playlist creation system. With the AST model being pre-trained on large, diverse audio dataset 'AudioSet', this pretraining focuses on general-purpose audio features that span a wide variety of sound categories such as animal sounds and environmental sounds. For tasks that require a more granular understanding—like distinguishing between closely related subgenres of Electronic Dance Music—fine-tuning on a specialized dataset is essential.

Fine tuning is done by attaching a fully connected classification head, with the size of the number of genres, as the output layer to train the model effectively. While the goal of this system is not to create a genre classification system, fine tuning it this way is essential to train the model to learn small differences between closely related genres.

Subgenres within EDM, such as Afro House, Melodic House, and Progressive House, often share extremely similar overlapping characteristics, such as tempo, instrumentation, and structure. But they also have subtle differences in rhythm patterns, harmonic progressions, and sound design.

These small differences are often imperceptible to models trained on broad datasets but are vital for accurate classification and ultimately playlist generation. After the model has been fine-tuned to understand subgenre-specific differences, the vector embeddings become more informative and discriminative. This improved quality of the embeddings is crucial for the next step in the pipeline: computing cosine similarity.

3.6 Cosine Similarity and Playlist Generation

Cosine similarity measures the angle between two vectors (A and B) in a high-dimensional space, providing a metric for how similar two embeddings—and thus two songs—are. The effectiveness of this metric depends on the embeddings being rich in meaningful features.

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

By fine-tuning the AST mode to learn differences between the subgenres, the embeddings therefore inherit the learnings as well. This ensures that the computed cosine similarity scores between the embeddings are highly indicative of the musical compatibility between which. This ultimately allows the system to identify tracks that are truly similar to the user's query song. The system then returns the k-highest scoring cosine scores of all entries in the music database relative to the input song, which represents the end resulting playlist.

GitHub Link to all code:

https://github.com/ColinHoude/BetterPlaylists

4. Results

The training itself was simple and effective. A classifier layer with the number of genres was added as the output layer to the model and the model was trained using the following parameters:

- Cross Entropy Loss for Classification
- AdamW Optimizer
- Learning Rate Scheduler
- Batch size of 16
- Trained on P100 GPU
- 600+ songs using an 80/20 train test split

Training converged around a 0.1 loss value over five epochs which took around an hour and a half. The training and test loss over epochs graph is shown below with the test accuracy over epochs graph following:

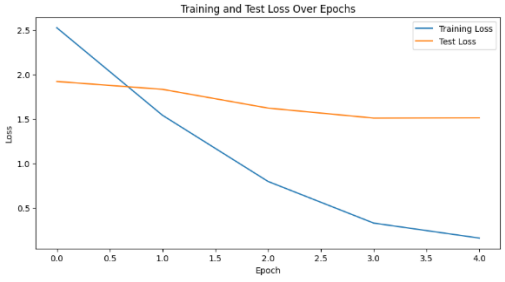


Figure above shows the training and test loss over five epochs

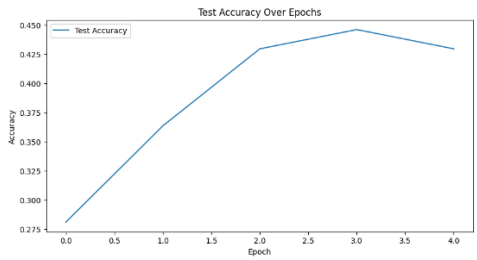


Figure above shows the test accuracy over five epochs

4.1 Genre Classification and Playlist Creation

After convergence, the model finished with a test accuracy of around 45%. While this is not an amazing score, the model did increase in score up from 28% pretraining. There are a few main reasons the model did not score high accuracy across the specified subgenres. The most prominent of which being the subgenres are simply just too similar. With all of these subgenres being under the 'Electronic Dance Music' umbrella, it can be very difficult to distinguish between them. Most of the have many overlapping characteristics and can be argued that one song could belong to not just one of the subgenres but many at once.

To make sure this was the case I fine-tuned a separate AST model on distinctly different music genres. I chose to the following four genres: Country, Hip Hop, Classical and Techno.

Using the same exact system, the same starting pre-trained AST model, the same training parameters, only changing the dataset for classification, I achieved much better results. When using this same system to classify genres across distinctly different types of music the model scored much a higher accuracy at 97.43% on a test set. The model also trained much quicker, converging to a loss of less than 0.001 after just two epochs.

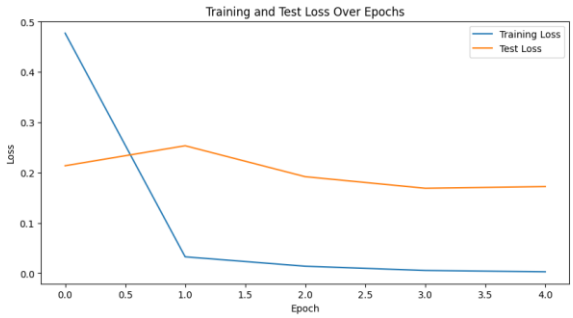


Figure above shows the training and test loss over five epochs

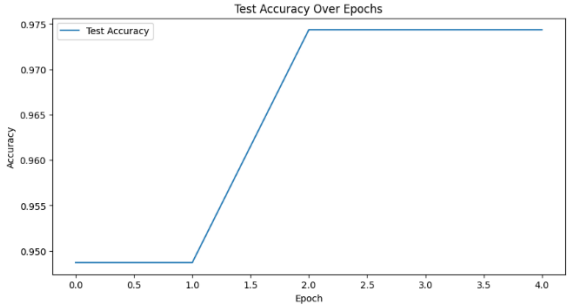


Figure above shows the test accuracy over five epochs

This shows that the system preforms well when classifying music across different genres and it is

simply the genres being too overlapping to score high accuracy in regard to the subgenres within the EDM umbrella.

But again, this is not a classification system. This is a playlist creation system. When using the fine-tuned model to create EDM playlists, the system performed very well. When using cosine similarity to return the k-closest songs, the system returned songs all with a similarity score of 94% and higher. Below is an example output of a test run where the input song is the song being requested to make a playlist around and the recommended songs are the songs that would make up the playlist:

- Input Song: Namtso (Extended Mix).mp3 Recommended Song: Namtso (Extended Mix).mp3, **Similarity: 1.0000**, Genre: Downtempo
- Recommended Song: Volen Sentir Makebo Alchemist (Original Mix).mp3, **Similarity: 0.9541**, Genre: Downtempo
- Recommended Song: The Temper Trap - Sweet Disposition (John Summit Silver Panda Remix).mp3, **Similarity: 0.9473**, Genre: Melodic House
- Recommended Song: Breathe (Extended Mix).mp3, **Similarity: 0.9463**, Genre: Progressive House
- Recommended Song: PROFF Volen Sentir - The Rumble - Original Mix.mp3, **Similarity: 0.9461**, Genre: Downtempo

5. Conclusion and Future Work

This project demonstrates the versatility and potential of the Audio Spectrogram Transformer (AST) in the realm of music genre classification and playlist generation. Fine tuning the AST model for genre classification has shown to be a good avenue to produce new music playlists solely on the audio representation itself. It was interesting to use the embeddings for an unintended task to produce a system that does something completely different than what is intended.

This system still has a lot of room to grow. This system could easily be adapted to larger scale databases with more music to better train the model to enhance its playlist creation abilities. While I only adapted the system across a small number of niche genres, it has shown the effective use case, and I am confident it can be used effectively at a much larger scale.

6. References and Links

[1] - Gong, Y., Chung, Y.-A., and Glass, J. 2021. AST: Audio Spectrogram Transformer. *arXiv preprint arXiv:2104.01778* https://arxiv.org/pdf/2104.01778

[2] - Sara Atito Muhammad Awais Wenwu Wang Mark D Plumbley Josef Kittler, 2022. ASiT: Audio Spectrogram vIsion Transformer for General Audio Representation https://arxiv.org/pdf/2211.13189v1

[3] - https://github.com/crlandsc/Music-Genre-Classification-Using-Convolutional-Neural-Networks

[4] - Craus, Andreea C.; Berger, Ben; Hughes, Yves; and Horn, Hayley () "Leveraging Transformer Models for Genre Classification, "SMU Data Science Review: Vol. 8: No. 1, Article 1. https://scholar.smu.edu/datasciencereview/vol8/iss1/1

GitHub Link to Project:

https://github.com/ColinHoude/BetterPlaylists