Investigating Audio Encoding Models' Effectiveness on Downstream Tasks

Caroline Cahilly, Caleb Liu, Thomas Yim
Department of Computer Science, Stanford University
ccahilly@stanford.edu, calebliu@stanford.edu, yimt@stanford.edu

1 Introduction

Everyone has music that speaks to them. An Alzheimer's patient hears a song from their youth and momentarily remembers their granddaughter. NBA player Jaylen Brunson listens to Justin Bieber before every game he plays. Recommendation systems are just one example of the many ML-based music systems that will continue to define how we create and consume music. It is paramount to have robust audio embedding systems to represent music in a way that ML systems can understand/use.

Reasons that people are drawn to a piece of music range from instrumentation, to the timbre of voice, to simply the "vibe". The core challenge lies in extracting semantically rich audio embeddings from raw audio, enabling more accurate and context-aware recommendations that take into account all the factors of music. The input to our algorithms is raw music audio. We then employ audio embedding frameworks such as MERT, Wav2Vec 2.0, and CLAP to produce high-level vector representations (embeddings) of these audio signals, and use them on two tasks: genre classification (multi-class classification) and music captioning (text generation). These tasks serve as proxies to evaluate how well different embeddings capture musically relevant information. By comparing the performance of multiple embedding techniques on these downstream tasks, our work aims to identify the most effective methods for generating rich, semantically meaningful audio representations.

2 Related Work

The effectiveness of audio embeddings for music-related tasks has been a subject of growing research, which inspires our evaluation of MERT, Wav2Vec 2.0, and CLAP embeddings in genre classification.

Ding and Lerch (2023) study how four audio embedders (VGGish, OpenL3, PaSST, PANNs) can be used as teachers for knowledge transfer in music classification tasks. They show that music-specific embeddings can significantly improve performance on downstream tasks, improving classification accuracy by up to 15 percent compared to models trained without knowledge transfer.

Lastly, Perera et al. (2024) explored the use of contrastively learned audio embeddings, specifically the L3-Net Model, for various audio classification tasks such as music tagging, speech detection, and environmental sound recognition. Their findings revealed that these embeddings outperformed traditional hand-crafted features such as MFCCs and spectral descriptors. However this model uses a convolution based encoder and there have been better transformer-based pretrained models that might work better that we want to explore.

3 Dataset and Features

The dataset we used for genre classification was the GTZAN dataset (Olteanu, 2023). This contains 1000 audio clips (each 30 seconds) evenly distributed across 10 genres (blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, rock). This was split into 800 training and 200 test examples.

The dataset we used for generating a description was MusicCaps (Agostinelli et al., 2023). We had 5,360 total examples, with a train-val-test split of 70-10-20. Each example was a pair of (i) a 10-second song clip from YouTube and (ii) a descriptive text caption written by a musician describing the song. The descriptions include information about the instruments, emotion, audio quality, sequencing

of their piece, etc. See the following caption as an example: *This is a Hindustani classical music piece. There is a harmonium playing the main tune...*

We pulled the song clips as WAV files from YouTube, then resampled to the required sampling rate depending on the embedding model used (CLAP: 48khz, Wav2Vec 2.0: 24khz, MERT: 24khz). Some of the songs were stereo (two-channel audio with spatial differentiation) and others were mono (single-channel audio); we converted all tracks to mono to ensure consistency in audio format for processing. We also normalized the audio to the range of [-1, 1] to ensure consistent amplitude levels across all samples to prevent distortion.

4 Methods

4.1 Embedding Methods

For both tasks, we used three different audio embedding methods. We ran experiments where we used the pre-trained embeddings "as-is" for our downstream tasks. We also ran experiments where we unfroze the embedding model weights to fine-tune them alongside the model used for the downstream task (see below). The base embedding models we used were the following:

1. **Contrastive Language-Audio Pretraining (CLAP)**: CLAP (Wu et al., 2024) learns audio representations via contrastive learning on a dataset of general audio clips paired with text descriptions. For each example i, the model computes an audio embedding E_i^a by passing audio a_i through a convolutional audio encoder and an MLP, and a text embedding E_i^t by passing text t_i through a text encoder and an MLP. The model is trained with a contrastive loss (1) that pulls ground-truth audio-text pairs closer and pushes all other pairs apart. Here, τ is a learnable temperature parameter, and N is the batch size:

$$L = \frac{1}{2N}\sum_{i=1}^{N}\left(\log\frac{\exp(E_i^a \cdot E_i^t/\tau)}{\sum_{j=1}^{N}\exp(E_i^a \cdot E_j^t/\tau)} + \log\frac{\exp(E_i^t \cdot E_i^a/\tau)}{\sum_{j=1}^{N}\exp(E_i^t \cdot E_j^a/\tau)}\right) \tag{1}$$

2. **wav2vec 2.0**: wav2vec 2.0, designed by Baevski et al. (2020), differs from CLAP in that it learns audio represntations direclty from raw speech audio using a self-supervised aprpaoch, and is then fine-tuned on a small set of labeled data. During pre-training, raw audio is first processed through a multi-layer convolutional encoder, producing latent features $z_1, ..., z_T$ for the T time steps. A transformer then computes contextualized representations $c_1, ..., c_T$, which capture sequence-wide information. Simultaneously, a quantization module discretizes the latent features $z_1, ..., z_T$ into q_t. For some masked timestep t, the contrastive loss is designed to maximize the similarity between c_t and the true quantized representation q_t, while minimizing similarity to negative samples $\tilde{q} \in Q_t$. Cosine similarity $sim(x, y) = \frac{x \cdot y}{||x||||y||}$, a temperature parameter κ, and a diversity loss term L_d (scaled by α) are used in the loss function:

$$L = -\log\frac{\exp(sim(c_t, q_t)/\kappa)}{\sum_{\tilde{q}\sim Q_t}\exp(sim(c_t, \tilde{q})/\kappa)} + \alpha L_d \tag{2}$$

3. **Acoustic Music Understanding Model (MERT)**: Li et al. (2024) uses a masked langauge modeling framework and incorporates both acoustic and musical-pitch "teacher" models to provide pseudo-labels for training, which enables handling of both tonal and harmonic properties of music. The acoustic teacher is based on a Residual Vector Quantisation-Variational AutoEncoder and discretizes the raw waveform into audio codewords, and the musical teacher provides pitch-related guidance through the Constant-Q Transform (CQT). The loss function for acoustic, L_H, is a classification-style loss derived from HuBERT's Hsu et al. (2021) prediction objective, and the loss function for musical-pitch, L_{CQT}, is a mean squared error reconstruction loss over masked CQT frames. Total loss in MERT is given by the linear combination of the acoustic loss function and the musical-pitch loss function.

4.2 Genre Classification Methods

We preprocess the audio then extract its embedding from the model. For MERT, the embedding dimensions are (time steps, 1024) so we take the mean across the time dimension. For CLAP, the

embedding dimensions are (512). For Wav2Vec 2.0, the dimension embeddings are (time steps, 768) so we also take the mean across the time dimension leaving us with (768). Then we project each of these embeddings to a hidden state of 512, apply ReLU, do batch normalization, then project to a dimension 10 vector that indicates how likely each genre is. We used cross entropy loss (3) .

4.3 Text Generation Methods

We preprocess the audio and extract its embeddings using the embedding models discussed above. We then inputted these embeddings directly into T5, a sequential model originally designed for text-to-text tasks (Raffel et al., 2023).

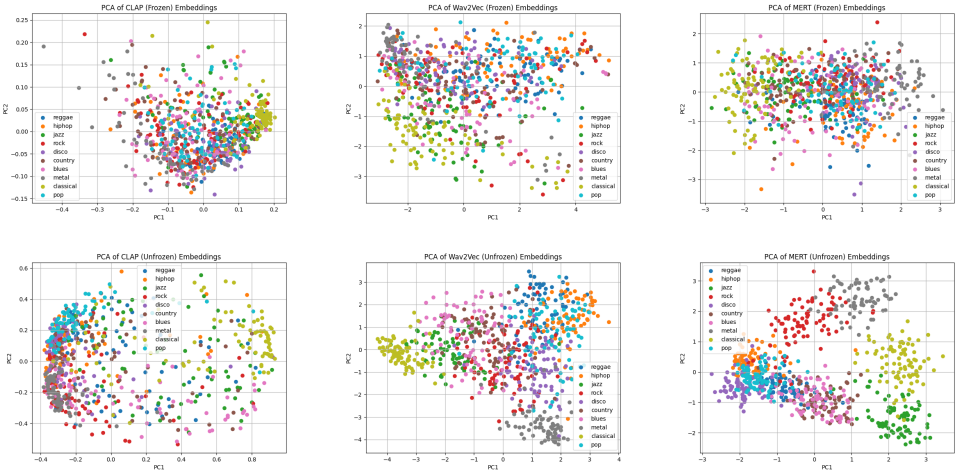
T5 expects input embeddings of shape (sequence length, 512). CLAP's audio embeddings were shape (512), so we fed them straight into T5 as (1, 512). Wav2Vec 2.0 outputs have shape (time steps, 768), so we used a linear layer to project them to size (time steps, 512) and then fed them to T5. MERT embeddings have shape (13, time steps, 768), where 13 is the number of hidden layers, and the embedding has one audio representation for each layer in the model. Given that each layer's representation performs differently in different downstream tasks, we decided to use a learnable weighted average representation to combine the layers. Specifically, we used a 1D convolutional layer to go from 13 channels to 1 channel, with a kernel size of 1. This gave us an embedding of size (time steps, 768). We reduced the time dimension then used a linear layer to project this embedding to size (time steps, 512), which we inputted to T5.

The loss function for T5 during fine-tuning is cross-entropy loss, where the model is trained to minimize the difference between predicted and actual token sequences. For a given batch of data of size N and sequence length T:

$$L = -\sum_{i=1}^{N}\sum_{j=1}^{T} y_{ij}\log(\hat{y}_{ij}) \tag{3}$$

where y_{ij} is the ground truth (one-hot encoded) and $\hat{y}_{ij}$ is the predicted probability for the j^{th} token in the i^{th} sequence. For our optimizer, we use Adam optimizer. Adam combines the benefits of momentum, which accelerates gradient descent by keeping track of the gradient's past directions to smooth updates, and RMSProp, which scales the learning rate adaptively for each parameter based on the magnitude of past gradients.

5 Experiments/Results/Discussion



(a) CLAP (b) Wav2Vec 2.0 (c) MERT

Figure 1: Top row is PCA on the train set embeddings before training the encoding model. Bottom row is after unfreezing its weights and training.

5.1 Genre Classification

5.1.1 Experiment Setup

There were six experiments run with each combination of Wav2Vec 2.0 to MLP, CLAP to MLP, and MERT to MLP with one experiment freezing the encoder parameters and the other training them. This is trained for 10 epochs with cross-entropy loss and the Adam optimizer. We used a learning rate of 1e-5 as 1e-4 did not have a steadily decreasing loss and 1e-6 was not converging quick enough. We are measuring performance as the accuracy at choosing the correct genre from the 10 options.

5.1.2 Results and Discussion

Embedding model	Train Accuracy (%)	Test Accuracy (%)
Wav2Vec 2.0 (Frozen)	44.25	43.50
CLAP (Frozen)	37.00	36.63
MERT (Frozen)	72.12	66.50
Wav2Vec 2.0 (Unfrozen)	92.75	67.50
CLAP (Unfrozen)	63.75	43.00
MERT (Unfrozen)	100.00	84.00

Table 1: Performance on genre classification task

As we can see in the table above, the best performing setup is the unfrozen MERT to MLP with a test accuracy of 84% where we are finetuning all parameters. In both situations where the encoder is frozen and unfrozen, MERT performs the best, then Wav2Vec 2.0, then CLAP. This makes sense because MERT is pretrained on music data whereas Wav2Vec 2.0 is trained on speech data and CLAP on general audio data. We also hypothesize that Wav2Vec performs better because it retains the temporal dimension while CLAP does not. Unfreezing the embedding model's weights results in much better performance across the board.

We investigated the diversity of the audio embeddings output by each model to get a heuristic for how separable the embeddings are (which would be useful in both classification and generation). We computed the embeddings for each model for all training examples and ran PCA 1.

As you can see in figure 1, MERT Unfrozen separates the classes into much more distinct clusters after finetuning occurs which explains its much higher performance. Although it is difficult to see in the graph, the MERT Frozen graph has completely distinct metal and classical separations where as there is overlap in CLAP and Wav2Vec. Before training, CLAP has a 97.1% average pairwise cosine similarity on all the training examples in our genre classification dataset which make it difficult to distinguish between them. The ability to separate the classes influences its classification performance.

5.2 Text Description Generation

5.2.1 Experiment Setup

We ran six total experiments for text generation. We created three pipeline models: Wav2Vec 2.0 to T5, CLAP to T5, and MERT to T5. We trained each of them with two settings: one where the embedding model was frozen, and another where the embedding model was fine-tuned alongside T5 (and the aggregation/projection layers if applicable). We used a learning rate of 10^{-4}, which we empirically found worked best, and the largest batch size compatible with our GPU (4 for Frozen Mert; 8 for all others). We trained each model for 15 epochs, and chose the model that achieved the best performance on our validation set.

5.2.2 Evaluation Metrics

For our qualitative analysis, we analyzed the raw text generated, comparing it to the ground truth descriptions and the audio itself. For our quantitative analysis, we used BERT similarity. BERT similarity is the average cosine similarity between BERT embeddings, where BERT is a pre-trained bidirectional transformer model for NLP tasks, such as text classification, question answering, and text similarity. Since we are looking at semantics (not a task like translation), we believe that BERT similarity is the best metric.

5.2.3 Results and Discussion

We picked the checkpoint that had the lowest loss on the validation set as our model that we would then evaluate on the test set (15-epoch model for all models except Wav2Vec 2.0 unfrozen, mert frozen, mert unfrozen, for which we used the 14-epoch model).

	Average BERT Similarity		
Embedding Model	Train	Val	Test
Wav2Vec 2.0 (Frozen)	0.6158	0.6124	0.6137
CLAP (Frozen)	0.6318	0.6349	0.6282
MERT (Frozen)	0.6193	0.6306	0.6149
Wav2Vec 2.0 (Unfrozen)	0.6121	0.6205	0.6107
CLAP (Unfrozen)	0.5940	0.5895	0.5892
MERT (Unfrozen)	0.5544	0.5600	0.5536

Table 2: Performance on text description generation task across Train, Val, and Test datasets

We see that the BERT similarities for all models are similar, low values, suggesting that the generated captions are semantically different from the true captions. The similarity in values suggests that T5, not the embedding model, is the determining factor in the quality of the generated captions.

The best performer on the test data was frozen CLAP, and the worst by a significant margin was unfrozen MERT. We also see that the unfrozen models are almost always worse than the frozen models. We hypothesize that this is because the pre-trained embeddings, when left unchanged, better preserve the rich and generalized semantic information learned during pretraining on a diverse dataset. Perhaps this would be different if we used a different sequential model meant for audio inputs.

Qualitatively, we see that the semantic similarity between the generated and true captions is quite low. Consider the following example output from frozen and unfrozen wav2vec 2.0. This analysis is analogous to what we see with CLAP and MERT.

1. **True:** The low quality recording features a live performance with an electro song playing in the background and crowd cheering sounds...
2. **Generated (frozen):** The e-guitar is a reverberant and reverberant e-guitar. It is a bit noisy and in mono.
3. **Generated (unfrozen):** The low quality recording features a live performance of a saxophone solo melody. It sounds passionate, emotional and passionate. (Note that the all other examples we sampled also had this caption.)

We see that the generated captions are song descriptions, but not necessarily for the correct song. The captions often repeat words/phrases like in the frozen generated caption. There also may be some semantic similarity, such as with the true caption and the generated frozen caption. Additionally, for some of the models, such as the wav2vec 2.0 unfrozen model, we see that we get the same generated caption many times for different ground truths. The qualitative results also suggest issues with the T5 model, which we hypothesize is because T5 expects text (not audio) embeddings as inputs.

6 Conclusion/Future Work

Our study highlights the strengths and limitations of MERT, Wav2Vec 2.0, and CLAP embeddings in the downstream tasks of genre classification and text generation, with MERT out performing the others on genre classification, likely due to its training on music-specific tonal and harmonic properties. To further advance this work, we would investigate whether the new embeddings trained on genre classification would transfer well to more nuanced benchmarks such as instrument classification, text captioning, or vocal-tone or accent recognition. We assume that some tasks might better leverage the nuanced strengths of each embeddings, such as Wav2Vec 2.0's raw audio-trained embeddings or more general-purpose embeddings with CLAP. With generating a text description, we believe there could be better performance if we had more compute and could train for more epochs or use a larger LM like Llama or GPT to handle the embedding to text translation.

7 Contributions

- **Caroline Cahilly**: Dataset preparation, text generation models, report writing, attempt at DPO model and speech2text model (not discussed in report)
- **Caleb Liu**: Dataset preparation and preprocessing, genre classification models, error analysis, milestone report writing
- **Thomas Yim**: Audio to text implementation, genre classification models, PCA embedding analysis, evaluation setup, milestone report writing, attempt at finetuning MusicGen model (not discussed in report)

References

Andrea Agostinelli, Timo I. Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, Matt Sharifi, Neil Zeghidour, and Christian Frank. 2023. Musiclm: Generating music from text.

Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. 2020. wav2vec 2.0: A framework for self-supervised learning of speech representations.

Yiwei Ding and Alexander Lerch. 2023. Audio embeddings as teachers for music classification.

Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. 2021. Hubert: Self-supervised speech representation learning by masked prediction of hidden units.

Yizhi Li, Ruibin Yuan, Ge Zhang, Yinghao Ma, Xingran Chen, Hanzhi Yin, Chenghao Xiao, Chenghua Lin, Anton Ragni, Emmanouil Benetos, Norbert Gyenge, Roger Dannenberg, Ruibo Liu, Wenhu Chen, Gus Xia, Yemin Shi, Wenhao Huang, Zili Wang, Yike Guo, and Jie Fu. 2024. Mert: Acoustic music understanding model with large-scale self-supervised training.

Andrada Olteanu. 2023. Gtzan dataset - music genre classification. Accessed: 2024-12-06.

Dushani Perera, Maneesha Rajaratne, Shiromi Arunathilake, Kasun Karunanayake, and Buddy Liyanage. 2024. A critical analysis of music recommendation systems and new perspectives. *University of Colombo School of Computing*. Critical survey and hybrid recommendation proposal.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2023. Exploring the limits of transfer learning with a unified text-to-text transformer.

Yusong Wu, Ke Chen, Tianyu Zhang, Yuchen Hui, Marianna Nezhurina, Taylor Berg-Kirkpatrick, and Shlomo Dubnov. 2024. Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation.