Latent Feature Extraction for Musical Genres from Raw Audio

Arjun Sawhney, Vrinda Vasavada, Woody Wang
Department of Computer Science
Stanford University
sawhneya@stanford.edu, vrindav@stanford.edu, wwang153@stanford.edu

Abstract

This paper proposes and evaluates preliminary models to produce musical style encodings with applications in music style transfer. Inspired by methods of neural style transfer [7], we seek to learn encodings of musical style directly from raw audio data. We evaluate our models primarily qualitatively in their ability to obtain interpretable embeddings of musical genre, which we hypothesize will be strongly correlated with musical style. Additionally, we also benchmark our models quantitatively based on precision, recall, and F1 scores on a genre classification dataset. For our final model, we propose a hybrid encoding and classification approach (with an adapted loss function), which obtains visually promising 64-dim and 4-dim encodings of musical genre and achieves upwards of 94% and 65% accuracy on our genre classification train and test sets, respectively.

1 Introduction and Task Definition

With the success of neural style transfer [7], there has been an increasing number of attempts at performing music style transfer. Unlike in images, however, style is not as well defined for music. Intrinsic properties such as timbre and rhythm alone may not encapsulate what defines a song's style; however, the genre of a piece of music is highly related to its stylistic properties, which makes it particularly important in the field of music information retrieval (MIR). While both genre classification and musical style encoding are tasks that have been attempted, much of the work in those contexts involves extensive feature engineering.

In this paper, we consider musical genre to be directly correlated with style, and as such, attempt to learn a latent representation of it (using both supervised and unsupervised learning methods) directly from raw input audio. Concretely, we investigate hybrid neural networks with both autoencoding and classification components to learn genre embeddings. We evaluate our results primarily with the feasibility and interpretability of our embeddings when visualized using PCA. Additionally, we also look at classification metrics such as precision, recall, accuracy, and model error to benchmark our models.

2 Related Work

We primarily draw inspiration from previous work in neural style transfer for images. In neural style transfer, a common method of extracting a meaningful representation of style in an image is to use intermediate layers of a pretrained image classification network, such as the VGG-19 [7]. Thus, in our task of learning style encodings of music, we initially seek to train a music genre classifier in hopes that intermediate layers in the network will have a meaningful representation of musical style.

With regards to the task of music genre classification, we are motivated by promising work done by Tzanetakis et al. on the GTZAN dataset [6]. Based on examination of multiple previous works, we see that classification accuracy decreases significantly with an increase in the number of genres. Furthermore, since our work is primarily focused on learning potential style encodings of music, we select a subset of the original dataset to work with, namely the four genres of classical, jazz, pop, and metal. Finally, while we have seen precedence in approaches to genre classification using significant manual feature engineering in transforming inputs to Mel-Frequency Cepstrum Coefficients (MFCC) and Mel-Spectograms, we seek to experiment with learning musical style encodings directly from raw audio data [2, 3, 4].

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

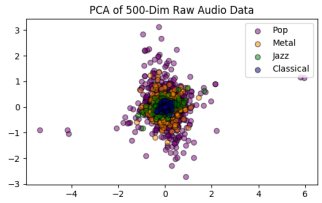

Figure 1: Visualization of PCA on raw input data

3 Dataset and Features

We started with the GTZAN Genre Collection Dataset which contains 1,000 tracks (each 30 seconds long) of 10 genres [1]. As discussed in Related Work, due to our priority of learning interpretable style encodings, we chose to use only the 4 genres of classical, jazz, pop, and metal. First, we converted these files into .wav format and used the Python library LibROSA to convert the audio files to a raw audio time series of amplitudes. We then augmented our dataset by splicing each song into one second segments.

With a native sampling rate of 22.05 kHz, each of the original samples was in $\mathbb{R}^{20,000}$, so we used average pooling with a pool size of 40 to downsample the dimensionality of our data, which doubles as a regularization technique. We ended up with an equal number of the four genres and 8000 examples in total. Each example was represented as a vector in $\mathbb{R}^{500}$. We chose a random split of 6000-1000-1000 for our train, development, and test sets, respectively.

Since our task was to learn encodings from raw data, we did not use any explicit feature engineering. Figure 1 shows a visualization of the raw data using PCA (a variance-maximizing dimensionality reduction algorithm), in which the genres are clearly not distinguishable from each other.

4 Models and Method

4.1 Two Layer Neural Network

For a classification model, we initially implemented a basic two layer neural network with one hidden layer in $\mathbb{R}^{128}$ and tanh activation. Our loss for one example is defined as

$$\mathcal{L}_{cross-entropy} = -\sum_{j=0}^{3} y_j \; log(\hat{y_j}) \tag{1}$$

where $y \in \mathbb{R}^4$ is a one-hot vector with a one in the component corresponding to the true class, and $\hat{y} \in \mathbb{R}^4$ represents the output of our classifier.

4.2 Vanilla Autoencoder

We then implemented a vanilla autoencoder with a single hidden layer in the encoder and decoder as a baseline. However, this was extended to a deeper architecture as seen in the top half of Figure 2 for a more fair comparison to the final model. We proceed to reference this model as a vanilla autoencoder in the rest of this paper. With such models, we seek a useful latent representation of the input audio $x \in \mathbb{R}^{500}$ by attempting to learn $f : \mathbb{R}^{500} \rightarrow \mathbb{R}^{64}$ and $g : \mathbb{R}^{64} \rightarrow \mathbb{R}^{500}$ where $f(x) = z$ for some $z \in \mathbb{R}^{64}$, and $g(f(x)) \approx x$. Note that f is the encoder and g is the decoder - both modeled as neural networks. Our training objective for any autoencoder is to minimize the reconstruction loss of recovering the original input when passed through the encoder-decoder pair. Numerically, this is defined as

$$\mathcal{L}_{reconstruction} = ||x - g(f(x))||_2^2 \tag{2}$$

4.3 Deep Softmax Autoencoder

In our final model, we combine the two approaches, using the result of the encoder as input to a multi-class classifier to form, what we call, a Deep Softmax Autoencoder. We theorize that this approach may reduce overfitting (in the classification component) because the classifier takes as input a vector in $\mathbb{R}^{64}$ instead of $\mathbb{R}^{500}$. To account for the combined model, we modify our objective to minimize a weighted combination of reconstruction and softmax cross-entropy loss aforementioned. This is formally defined for one example in Equation 3.

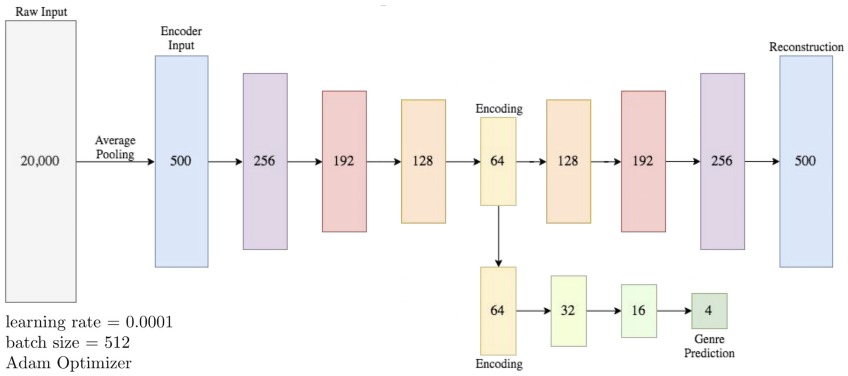

Figure 2: Model architecture of combined deep autoencoder and feed-forward multi-class classifier, referred to as a Deep Softmax Autoencoder

$$\mathcal{L}_{reconstruction} = \gamma||x - g(f(x))||_2^2 - (1-\gamma)\sum_{i=0}^{3} y_i \, log(\hat{y}_i) \tag{3}$$

By encouraging the model to minimize reconstruction loss along with classification loss, the model should be more likely to learn a latent representation of genre while retaining important information to reconstruct the original piece of music. Intuitively, for both reconstruction and classification loss to decrease, the encodings must both represent the original input and encode some information about its genre.

Methodologically, upon settling on this blueprint approach, we ran consistent experiments to tune our hyperparameter values, such as the number of layers and the layer sizes in our final model. These, along with our final architecture, are reflected in Figure 2. Additionally note that in running our experiments, we decided to use a final value of $\gamma = 0.9$ for our modified loss function in order to more heavily weight reconstruction relative to classification.

5 Results and Discussion

We divide our evaluation into quantitative and qualitative metrics. We focus on measuring the performance of the Deep Softmax Autoencoder architecture through precision, recall, and F1 scores. In the qualitative analysis, we visualize potential 64-dim and 4-dim embeddings using PCA and discuss their benefits and tradeoffs.

5.1 Quantitative Analysis

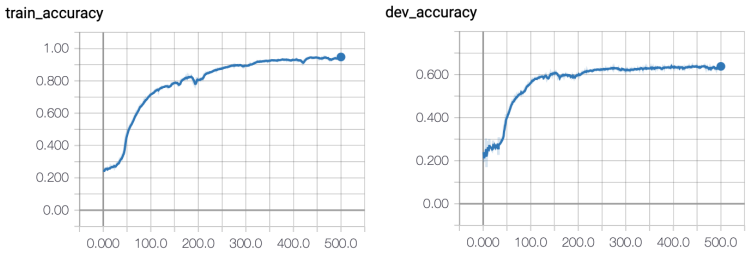

Figure 3: Deep Softmax Autoencoder accuracy curves with the epoch number on the x-axis

Classification Accuracies	Train Set (6000 Examples)	Dev Set (1000 Examples)	Test Set (1000 Examples)
Two Layer Neural Network	52.0%	38.1%	36.4%
Deep Softmax Autoencoder	94.9%	64.1%	65.3%

Table 1: Comparison of classification accuracy between Deep Softmax Autoencoder and baseline two layer neural network

Deep Softmax Autoencoder	Precision	Recall	F1 Score
Classical	0.783	0.775	0.779
Jazz	0.606	0.627	0.616
Metal	0.515	0.554	0.5337
Pop	0.670	0.608	0.638

Table 2: Objective metrics over a held-out test set of 1000 examples for the Deep Softmax Autoencoder

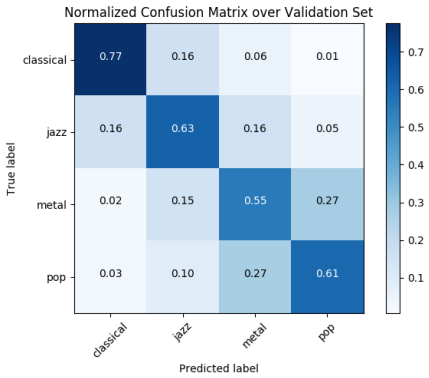

Figure 4: Confusion matrix of the Deep Softmax Autoencoder's predictions on a test set of 1000 held-out examples

From our baseline implementation of a basic two layer neural network as a genre classifier, we saw a relatively low training and test accuracy compared to previous works as described in Tzanetakis et al. [6]. As we increased the number of hidden layers in our classifier, we noticed a general trend of high overfitting. To combat this, we tried to reduce the dimensionality of our input (therefore reducing the number of weights in our network). This motivated our decision to use average pooling as a preprocessing technique to reduce dimensionality, as well as use dropout between layers with a final keep probability of 0.9 after tuning.

After implementing our Deep Softmax Autoencoder, we found a significant increase in training and test accuracy compared to the baseline two layer neural network. In addition, when examining the confusion matrix in Figure 4, we see that the entries are mostly concentrated along the diagonal, as desired. From Figure 4, the main sources of error are metal and pop pieces being mistaken for each other. We see that classical and jazz pieces are commonly mistaken for each other as well. From a human standpoint, these genres sound fairly similar, and we see further evidence to support their similarities in the visualizations of the latent spaces below. From analyzing our results in Table 2, we see that our final model obtains the highest precision, recall, and F1 score on classical music, which we hypothesize is due to classical music's more distinct style.

5.2 Qualitative Analysis

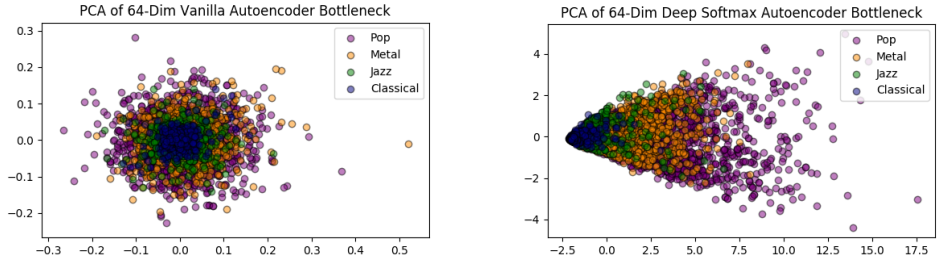

Figure 5: Visualization of PCA on bottleneck 64-dim encodings

In order to evaluate our encodings, we visualized them in 2-D space using PCA. First, we examined potential 64-dim encodings from the bottlenecks of the autoencoders we trained. In Figure 5, the vanilla autoencoder's results are as expected, since it is unsupervised and has no incentive to learn a distinguishable representation of genre. This is visible in the lack of separation in the latent space. We also visualized the Deep Softmax Autoencoder's encodings

when supervised with a genre classifier (architecture shown in Figure 2). These displayed promising separation and smoothness in the latent space. Qualitatively, we notice the jazz and classical Deep Softmax Autoencoder encodings are closely distributed in the latent space, which can likely be attributed to their similar instrumentation.

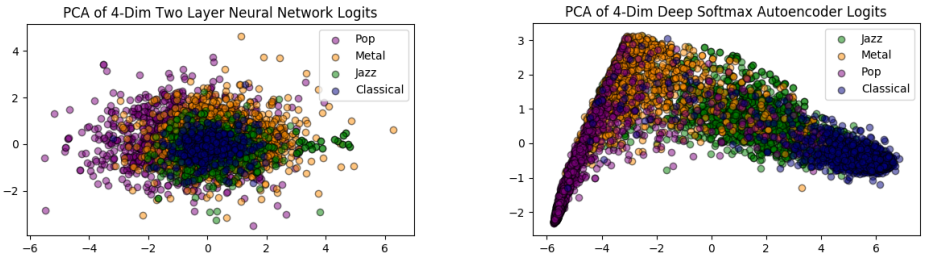

Figure 6: Visualization of PCA on 4-dim logits as potential encodings

Motivated by neural style transfer on images, we also visualized the classifiers' logits as another possible 4-dim genre encoding. As expected, due to the optimization objective, we see a clearer distinction between each class in the visualization of the classifiers' logits when accuracy is high. Again, like the bottleneck visualization, we see that the final model's logits, when used as encodings, display not only separation but also smoothness between clusters in the latent space. This is a desirable property for embeddings in general and less visible in the case of the two layer neural network logits, which seem to be inseparable clusters for all four genres in the latent space.

We notice in Figure 6 that our encodings for pop are not as clearly separable as the other three genres and are distributed with higher variance. As seen in the PCA visualization of the raw data in Figure 1, we observe a large variance in songs within the pop genre, which could explain the higher variance of the pop genre in the latent space. Upon listening to exclusively pop samples in the dataset, we found that pop songs seemed to have less of a distinct style compared to the other genres. We also observe a noticeable overlap between the pop class and the remaining three classes. When listening to random samples in the dataset, we found that pop songs could easily be mistaken for the other three genres, even by humans, which could explain the overlap in the latent space.

Compared to the 4-dim encodings, the 64-dim encodings have the potential to capture more subtle nuances within each genre. These encodings serve different purposes: particular tasks may require the expressivity of the 64-dim or the conciseness of the 4-dim.

6 Conclusion and Future Work

In conclusion, as shown in Figures 5 and 6, our attempt at learning genre embeddings purely from raw audio produces encouraging results for both 64-dim and 4-dim encodings. Our proposed hybrid model additionally outperforms our baseline model for genre classification, achieving around 95% and 65% compared to 52% and 36% accuracy on our train and hidden test sets for genre classification. We do notice, however, that in classification, our final model struggles to distinguish pop music whilst retaining strong performance on classical music. This is then reflected qualitatively in our embeddings (in both 4 and 64 dimensions), where pop music is more scattered as compared to the clustered classical music. In listening to and attempting to classify particular recordings ourselves, we posit that this discrepancy occurs due to the lack of a distinct style in pop music versus a clearer definition of classical music. Overall, our 64-dim embeddings display stronger granularity across genres, whilst our 4-dim embeddings indicate stronger separation. As aforementioned, both of these embeddings serve different purposes and will be useful in different scenarios.

In the future, we fundamentally seek to improve the interpretability of our latent representations. Specifically, we plan to experiment with using these encodings for musical style transfer and evaluate our embeddings in an extrinsic task. We also plan to interpolate components in our encodings to interpret the latent space. We acknowledge limitations in our approach, specifically in the trimming of the dataset and our avoidance of explicit feature engineering. As such, we hope to increase the number of classes in our dataset and expand our task brief to experiment with integrating MFCCs and other forms of feature engineering to see if we can further inform our encodings generated from raw audio. Finally, we are curious to see if replacing the autoencoder with a β-TCVAE would help us learn disentangled representations of genre via a mutual information gap (MIG) metric [10].

Contributions

Vrinda and Arjun worked on the initial data processing, which Woody then optimized. Vrinda and Woody worked on the initial classification model, and Arjun worked on the autoencoder. Vrinda and Woody worked on combining the two to form the Deep Softmax Autoencoder before we all collectively brainstormed ideas, ran experiments and evaluated results. We all worked on this report collectively.

References

[1] Music Analysis, Retrieval and Synthesis for Audio Signals (MARSYAS) GTZAN Dataset.

[2] H. Bahuleyan. Music Genre Classification using Machine Learning Techniques in arXiv, 2018.

[3] N. Mor et al. A Universal Music Translation Network in arXiv, 2018.

[4] I. Simon et al. Learning a Latent Space of Multitrack Measures in arXiv, 2018.

[5] S. Dai et al. Music Style Transfer: A Position Paper in arXiv, 2018.

[6] G. Tzanetakis et al. Musical Genre Classification of Audio Signals in IEEE, 2002.

[7] L. Gatys et al. A Neural Algorithm of Artistic Style in arXiv 2015.

[8] M. Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org, 2015.

[9] T. Li et al. "Automatic Musical Pattern Feature Extraction Using Convolutional Neural Network" in IMECS, 2010.

[10] R. Chen et al. "Isolating Sources of Disentanglement in VAEs" in arXiv, 2018.

Code can be seen here: `https://github.com/arjunsawknee/Genre-Extraction`