Predicting Song Moods: CS 7643

Julia Contarino, Blake Heimann, Anjana Shaji, Sara Srivastav
Georgia Institute of Technology
jcontarino3@gatech.edu, bheimann3@gatech.edu, ashaji6@gatech.edu, ssrivastav_31@gatech.edu

Abstract

The ability to classify songs has many applications, most notably in song recommendations. Common methods of classifying songs do not usually include the mood that a song evokes. This project aims to classify songs into four different moods - happy, sad, energetic, and calm songs. We use a combination of song features provided to us from Spotify's API which include acousticness, danceability, duration, energy, instrumentalness, etc. along with 30 second raw audio files. We analyzed multiple neural network architectures to make these classifications, including MLP, RNN, LSTM, and ensemble models. In this paper, we compare the different methods we explored and interpret our findings. In general, the neural networks trained solely on the extracted features performed better than the models trained solely on the raw audio files. The ensemble models varied in performance but ultimately performed slightly better than the best MLP trained on just the Spotify audio features.

1. Introduction/Background/Motivation

While Spotify has grown as one of the largest music subscription services, it offers pre-categorized playlists and recommendations of songs to its users, attempting to satisfy their innate desire to listen to songs related to how they are currently feeling. For instance, if a listener just went through a breakup, recommending upbeat love songs to that user would not be the ideal match, but sad songs may help offer some relief. Our aim is to use playlists labeled within Spotify to determine song moods with new inputs. This will assist song listeners to narrow down to the songs that more closely match the mood they are currently in and help them discover new music. This will result in better recommendations that will enhance the user experience in the music subscription industry. In most existing methods of music mood classification, songs are divided along the lines of energy and stress, from happy to sad and calm to energetic respectively. So we decided to stick with the most common four moods - happy, sad, energetic and calm. Currently there are neural networks that classify music genres using different models that work with either time-series data for the audio file or extracted audio features and lyrics, but few that leverage both. One other limitation to current practice would be people manually classify the songs to different moods or genres rather than making use of an automated system that classifies music. Our attempt is to model a neural network that combines both audio files and audio features in an ensemble. [12], [6], [3] discusses an ensemble model that combines both lyrics and audio files. [4] also does the same with music audio, lyric text, and social tags associated with music pieces. The results of this paper show that a model with a large data set of several mood categories made from combining lyrics and audio significantly outperformed systems using audio-only features [4]. Taking insight from [5] and [12], we experiment by training the models upon the songs' audio features supplied by Spotify and the raw mp3 files themselves. To further improve results, we ensemble various combinations of the models to most accurately predict the correct mood.

2. Approaches

To solve this problem, we created several models using preprocessed audio features from Spotify as well as raw 30 second audio files themselves. We created several models including the Multilayer Perceptron, CNN, RNN, and LSTM. We tested various epochs, batch sizes, and learning rates to optimize the models with the best hyper-parameters. The best parameters were then used to create an ensemble model. For our ensemble model, we combined the MLP model learned from Spotify features and either of the two recurrent neural networks learned from the raw audio to create a single network. We used the Adam optimizer and the Cross entropy loss as the loss function for calculating errors and performing gradient descent. Table 4 has the architecture used to design all our models.

2.1. Data Gathering and Preprocessing

To prepare our train, test, and validation datasets, we utilized Spotify's API to extract the audio features for the songs in each mood playlist. Approximately, 1600 songs were analyzed from 4 different mood categories. There

were 11 audio features for each track. These included confidence measures like acousticness, liveness, speechiness and instrumentalness; perceptual measures like energy, valence, and danceability; and descriptors like time-signature, duration, tempo, key, and mode.

A second dataset was extracted as a supplement to this original dataset. For each song already in our dataset, we extracted a raw 30 second clip of the mp3 from the Spotify API as well. However, each translated into a one-dimensional array of over 60,000 timesteps that would overburden any time dependent model with immense computation associated with backpropagation. To solve this problem, we converted the raw audio files into frequencies over each timestep by utilizing Mel Spectrograms in the librosa python library. Mel Spectrogram values break down the raw mp3 files into a reduced feature set that correspond to different frequency ranges over time frames [1] [13] [9]. Furthermore, we converted the raw value at each frequency to decibels, and leveraged the decibel values in timestep vs. frequency space. There were over 1000 songs that had raw audio files associated with them, and the Mel Spectrogram transformed the 1D array for each song into a two-dimensional array with approximately 1200 timesteps (0.25 seconds each) with 128 features (frequency ranges) consisting of floating point decibel values. These would serve as our preprocessed data from the mp3 files.

We normalized both of these datasets using MinMaxScaler. To solve the issue of class imbalance, we created a stratified train-val-test split to balance the number of samples for each mood label. We also provided class weights to the Cross Entropy loss function to manually rescale the weight given to each class. We determined the class weights by using Pytorch's WeightedRandomSampler class so that the models could place more emphasis on the minority moods. We also created our train dataloader with this weighted sampler as well so that our classifier model learns equally from all classes while training.

2.2. Multilayer Perceptron

In our first experiment, we solely used the Spotify provided audio features to train MLP neural network models to classify the songs into correct moods. The first MLP consisted of three linear layers each accompanied by batch normalization, ReLu activation function and a dropout layer, and a final output linear layer. The second MLP consisted of two linear layers each accompanied by a Hardtanh activation function, and a dropout layer followed by an output linear layer. The reason why we tried simple MLP models is that they all perform well on classification problems, where the given inputs are assigned a class or label and that our Spotify audio features are purely a tabular dataset.

2.3. CNN

The second approach we tried was a Convolutional Neural Network model. We wanted to see how CNN would classify non-image data like audio features. The model consisted of 2 sets of 1D convolution layers (3×3 kernel, padding = 1), batch normalization, ReLu activation, 1D convolution layer, ReLu activation, dropout layer and a max-pooling layer (3×3 kernel, stride = 2) followed by a final linear layer. CNN's learn filters similar to the spectro-temporal receptive field observed in the brain, which is the basis of the human auditory system. Therefore this NN architecture has the potential to classify music to specific moods in a similar way humans do so biologically [2], [7], [8]

2.4. RNN

A third approach our team explored was using the preprocessed audio data to classify songs into each mood using a Recurrent Neural Network (RNN). The benefit of using an RNN model as opposed to other models such as linear models or CNN is that RNNs capture the sequential aspects of data. Music is inherently time dependent, so extracting the sequential relationship of the decibel levels at each frequency over time is imperative. We fed these two-dimensional Mel Spectrogram arrays into the Pytorch multi-layer Elman RNN with tanh and a linear layer output that classified the results into the four moods. [11]

2.5. LSTM

The purpose of the next approach was to avoid the potential challenges associated with backpropagation and the likelihood of encountering an exploding or vanishing gradient in the RNN by leveraging an LSTM model and the advantages of its memory cell. The data preprocessing remained the same for the LSTM implementation as the RNN, using the Mel Spectrogram of decibel values in the frequency vs. time space. The implementation of the LSTM was a vanilla model, with architecture tweaks and various hyperparameter selections made during the training stage. The architecture changes consisted of the number of hidden layers, the number of LSTMs used back to back in a sequence of layers, as well as the linear layer size(s), and the number of linear layers used to generate the output to make the final mood prediction.

2.6. Ensemble

To create our ensemble model we used the Spotify audio features and the processed 30 second audio clips created in the models above. We tested multiple combinations of the models created above with an ensemble method in the hope that the combined sub-models would lead to more accurate final predictions than the independent models themselves.

Our ensemble model consisted of combining the multilayer perceptron solely trained on the Spotify features and either of the two recurrent neural networks - the RNN and LSTM model. To solve the problem of using two models with two related, yet different, datasets as the inputs, we created different data loaders with the same song index for each model input. We then passed in both datasets to the ensemble model which used the appropriate dataset to train the individual models before ensembling the models together. This overcame the challenge quite effectively given the shapes of the input data were of completely different dimensions for each model - the audio features being small 1D arrays of 11 features, and the preprocessed mp3 data being in the form of 128 frequency ranges by 1232 timesteps, each consisting of a decibel value (and a final shape of 128x1232).

Our first ensemble model consisted of concatenating the outputs of the audio feature model and the time series model so that each model's predictions contributed directly to the combined prediction. We then tried removing the last linear layer from both models and using the outputs of the second to the last layer as the inputs to the combined ensemble model. For both of these approaches, we loaded the independent models created above and linked them to the ensemble model. We froze the parameters for these pretrained models by setting requires-grad=False so that the gradients are not computed during backpropagation. We then added a classifier head on top of their concatenated outputs.

3. Experiments and Results

To measure our results, we split our initial data into training, validation and testing sets using stratification based upon the correct mood classification. Once we had trained our model on the training set, we then evaluated it on the data we held out in the test set. The accuracy of the testing data for each different approach is listed in Table 3. The per-class accuracy is show in Table 1. Accounting for any class imbalance in the training was handled at the model level for each model, rather than at the ensemble level.

3.1. Multilayer Perceptron

The two models implemented for the MLP were initially trained for 10 epochs, batch-size of 128 and learning-rate of 0.001. This gave us an accuracy of 77-78. Then the number of epochs was increased to 100 while reducing the learning rate to 0.0001. This resulted in a higher accuracy of 80. Further increasing the epochs resulted in the model not learning anything more. Also, changing the learning rates didn't help in improving the accuracy, nor did increasing the batch-size, as it had an adverse effect on the accuracy. In these cases, accuracy reduced to approximately 74-75 percent. With multiple sizeable linear layers, the number of parameters was determined to be sufficient for representing the problem. One interesting observation was that both the models had almost the same results across the different hyper-parameters tuned. The best accuracy we could get was with the first implementation, with details shown in Table 3. The model classified calm very well and struggled properly classifying between happy and energetic, as well as some between sad and happy.

Model	Calm	Energetic	Happy	Sad
MLP	1.00	0.73	0.67	0.82
CNN	0.99	0.72	0.71	0.80
RNN	0.98	0.06	0.89	0.07
LSTM	0.98	0.60	0.63	0.20
Ensemble RNN	0.98	0.85	0.67	0.70
Ensemble LSTM	1.00	0.92	0.64	0.76
Ensemble RNN (last layer removed)	0.98	0.83	0.70	0.70
Ensemble LSTM (last layer removed)	0.98	0.84	0.73	0.70

Table 1. Per class accuracy of the models.

3.2. CNN

In order to train the CNN model, we tried 10 epochs, batch-size of 128, and learning-rate of 0.001 initially, which gave us an accuracy of 77. We then increased the epochs to 100 and the learning rate to 0.0001, this resulted in an accuracy of 80. It was surprising to find that the CNN model commonly used for analyzing visual imagery, performed well on Spotify audio-features data. This model too classified calm very well, and struggled properly classifying between happy and energetic, as well as some between sad and happy. Since the MLPs are conventionally used for non-image tabular data classifications, we decided to go forward with the best MLP model for the ensemble model, which will be described in more detail in later sections.

3.3. RNN

While tuning the RNN model, we first tried a small number of epochs and gradually increased the amount until the accuracy on the testing set didn't seem to increase much with each addition compared to the additional training time the model required. We found that 100 epochs was the best balance of small training time to get high performance. Adding additional layers to the RNN model decreased the performance of the model given the 100 epochs we were running (to 0.46 testing accuracy), and that only one layer was sufficient to get the best results. When tuning the learning rate, we found that the RNN model did not demonstrate gradual learning at higher rates, between the range of 0.0001 and 0.01, achieving a testing accuracy between 0.50 and 0.56. We attempted lower rates and determined that the model performed best with an optimal learning rate

at around 0.00005. The best and final model achieved a testing accuracy of 0.57, which is the lowest performance among all the NN models explored. While the model classified almost all of the calm songs correctly and performed well on the happy songs as well, it struggled the most distinguishing sad songs, classifying them as calm or happy instead, and energetic songs, classifying them as happy instead.

3.4. LSTM

To train the LSTM, we attempted an assortment of values of the batch size, learning rate, hidden dimensions, linear layers, and their dimensions for the output, as well as the number of repeated LSTM layers in the model. For the batch size, we attempted batches of 128, 64, and 32. The lower batch size of 32 was first attempted due to GPU memory limitations until we switched machines to leverage a larger GPU with 16GB of memory. We also experimented with different numbers of epochs and learning rates, selecting 100 epochs as this gave the model plenty of iterations to update weights even at lower learning rates, without being overly time consuming, and aligning with prior experiments. The learning rates explored were originally in the range of 0.0001 to 0.001, but we continued to explore values between 1E-5 and 1E-4 as the loss curves indicated intense learning in the first couple of epochs. Using these reduced rates resulted in more gradual learning, with an optimal value of 3E-5. When exploring hidden dimension size, increasing the size of the hidden dimensions helped to offer more parameters and ensure the model was sufficiently complex to represent the problem. Furthermore, to allow the model the best convert these features into useful insights by passing through the linear layers to the output, we found that having an additional linear layer with a size commensurate with the hidden dimension improved accuracy. Therefore, a hidden dimension of 512 and linear output layers of 128, and 4 were used in the LSTM model. We explored adding multiple LSTM layers in the model, but this slowed training time and did not yield a noticeable improvement in accuracy. This final (best) model resulted in an accuracy of 66 percent, which was a slight improvement on the RNN but not as performant as the MLP. It classified calm very well but struggled to properly distinguish between happy and energetic, as well as between sad and happy. Interestingly, it did not have issues with differentiating between sad and energetic.

3.5. Ensemble

To determine the best parameters for the ensemble model, the optimal batch size, learning rate, hidden dimensions, linear layers and their dimensions for the output from the MLP, RNN, and LSTM created above were taken into consideration. The same hidden dimension and output layer dimension for the Recurrent Networks had to be used to create the ensemble model. Since we preloaded the weights for the submodels, we used a higher learning rate to create the combined models. We found that the best learning happened with 100 epochs, a batch size of 128, and a learning rate of 0.01. The ensemble model with the combined datasets outperformed the individual models. Removing the 2nd to the last layer of each network did not help in increasing the accuracies of our model. The LSTM ensembled models were slightly better than the RNN models. The testing accuracies for these combinations of these models can be seen in Table 3. The best performing model was the ensembled MLP with the Spotify audio features combined with the audio LSTM network.

The examination of learning curves for the simple LSTM ensemble model with no modifications to the submodels shows us that we needed fewer training samples to achieve the same or better classification accuracies than systems using just the Spotify audio features or audio Mel Spectrogram singularly. This learning and loss curve for this model can be seen in Figure 1. Over multiple epochs, we can see that the loss and accuracy curves do not vary. The training loss continues to decrease with experience but the validation loss remained stagnant. After 40+ epochs the model begins to overfit to the training set.

The confusion matrix for the LSTM ensemble model with the best accuracy can be seen in Figure 2 and the precision and recall for each mood classified can be seen in Table 2. This model was able to correctly classify the moods of 321 out of 378 songs. It achieved an accuracy of 85% with 512 hidden dimensions and 2 output layers. Looking at the confusion matrix and Table 2 for this model, we can see that this model was able to correctly classify all calm songs but had difficulty discerning between energetic and happy moods. Overall this LSTM ensemble model had the highest accuracy classifying the songs. It did better than the simple RNN and LSTM as well as the other combinations of the ensemble network attempted with the highest precision, re-

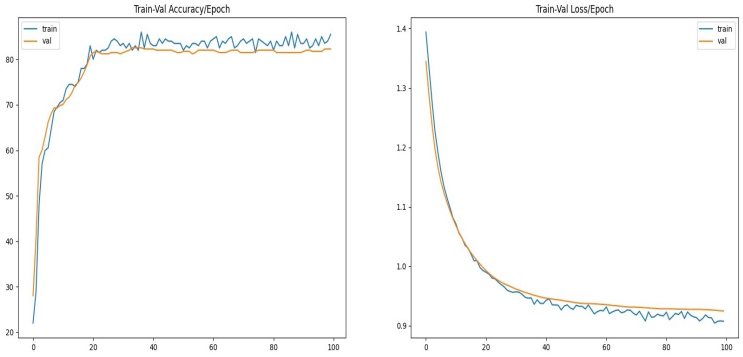

Figure 1. Accuracy and Loss of the LSTM Ensemble Model

Mood	Precision	Recall	F1-Score	Support
calm	1.00	1.00	1.00	126
energetic	0.75	0.92	0.82	89
happy	0.75	0.64	0.69	92
sad	0.86	0.76	0.81	71

Table 2. Precision and Recall of each mood for the LSTM Ensemble model.

Model	Learning Rate	Batch Size	Testing Accuracy
MLP	0.0001	128	0.80
CNN	0.0001	128	0.80
RNN	0.00005	128	0.56
LSTM	0.00003	128	0.66
Ensemble RNN	0.001	64	0.83
Ensemble LSTM	0.01	128	0.85
Ensemble RNN last layer removed	0.01	128	0.83
Ensemble LSTM last layer removed	0.01	128	0.84

Table 3. Model Hyperparameters, and Results

call, and f1-score for all moods. In comparison to the MLP and CNN models, this model performed better with higher precision, recall, and f1-score for calm and energetic songs and equally as well classifying sad and happy songs.

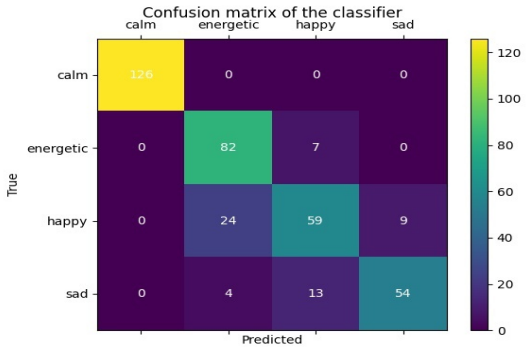

Figure 2. Confusion Matrix for LSTM Ensemble Model

4. Discussion

One of the challenges encountered with the recurrent neural network implementations was that any model with sufficient hidden dimensionality, as soon as the accuracy on the training set reached above 60 percent, the validation accuracy began to falter. This indicated that the model was overfitting to the training set, at a very early stage. With continued training for which the training accuracy increased further, the validation accuracy began to dip drastically, with only approximately 60 percent accuracy on the validation set for any given epoch. This had two implications, the first being the potential for overfitting, and the second suggesting that our models had a hard time generalizing the information learned from the training set to the validation. The first point was addressed by monitoring learning and strategically selecting learning rates. The second was likely due to the data preprocessing step for converting the raw audio to Mel Spectrogram, potentially causing the transformed data to lose information. This could be why our accuracy levels for the LSTM and RNN are drastically lower than the MLP or CNN.

Model	Architecture
MLP	linear layer 1: $11 \rightarrow 512$
	linear layers: $512 \rightarrow 128 \rightarrow 64 \rightarrow 4$
CNN	1D conv layer 1: $1 \rightarrow 40$
	1D conv layers: $40 \rightarrow 80 \rightarrow 160 \rightarrow 160$
	output linear layer: $320 \rightarrow 4$
RNN	128 hidden dim; 1 RNN layer
	1 linear layer out $512 \rightarrow 128 \rightarrow 4$
LSTM	1 LSTM Layer, 512 hidden dim
	2 linear layers out $512 \rightarrow 128 \rightarrow 4$
Ensemble RNN	128 hidden dim; 1 layer
Ensemble LSTM	512 hidden dim; 2 layer
Ensemble RNN last layer removed	128 hidden dim; 1 layer
Ensemble LSTM last layer removed	512 hidden dim; 2 layer

Table 4. Model Architectures

Another challenge was managing memory in the original GPU (2GB) for any model leveraging the Mel Spectrogram data. As the batch size increased, more memory was required and training had to leverage a larger GPU (16GB).

With higher learning rates, we observed more sporadic and volatile learning curves, with the majority of learning taking place in the early epochs. In this state, the validation accuracy fluctuated around an accuracy value as it got stuck in a local minima and the model stopped learning, or it over-fitted the training data as described above. Having a reduced learning rate led to more gradual learning and continuous improvement (though at a slower rate) in later epochs. Furthermore, the larger batch sizes seemed to better smooth the overall validation performance over each epoch as well. Additionally, maintaining sufficiently parameterized models (large enough hidden dimensions or linear layer sizes) proved to be important for ensuring the models could represent the problem well.

Overall, the results were promising, but at 85 percent accuracy, likely would not suffice as a production model, though we are confident that additional data, more extensive training, and additional feature engineering would help. One interesting outcome from analyzing the confu-

sion matrices of the models was which moods were the most challenging to accurately predict, and which moods were the common misclassifications. It was common amongst all models that the calm class was nearly perfectly classified, while significant variations in performance existed among the other mood classes. The misclassifications of moods had observable patterns in our better models, specifically our best ensemble model of the LSTM and MLP. Notably, the energetic and happy moods had a common overlap of misclassifications, indicating the challenge faced by the model in differentiating these two moods from one another. Likewise, a similar relationship was observed between happy and sad. However, this was not the case for sad and energetic and the model did a good job of differentiating the two. This is interesting because it indicates that there may be common features between energetic and happy, as well as sad and happy for a subset of songs which caused the model to make incorrect classifications. This could be attributed to the fact that the 30 second clip may not represent the entire song well. If the first 30 seconds of the song (our clip) more closely resembles the features of another mood, we would expect to get an incorrect classification.

5. Conclusion

To conclude, our ensembled model of the LSTM and MLP are sufficient for making a rough prediction for mood classifications with an accuracy of approximately 85 percent, based upon extracted audio features as well as spectrogram data from the raw audio. Classifying calm and energetic were greater than 90 percent accurate, while happy and sad classifications bordered on 70 percent accuracy. For making predictions for music classifications to offer enhanced song recommendations, this model is sufficient but could use improvement. As this machine-learned classification serves to provide value in entertainment and does not have any health or financial implications, a model of this robustness is arguably sufficient.

6. Further Considerations

As our dataset only consisted of 1600 English songs and only 4 moods, it would be interesting leverage more songs in different languages over more moods. With more data, our model could yield improved performance and be applied to more songs and many other moods.

Another important factor determining the outcome of these experiments had to do with the audio data and its preprocessing. Having the entire song rather than a 30-second clip would be an obvious improvement, and various transformations could have been leveraged instead of the Mel Spectrogram, such as using the original Spectrogram and not converting to decibels. Further, the python library librosa offers many audio processing capabilities beyond what we implemented in this paper. Additional feature extraction and transformation could be applied extensively using this library that may yield superior results.

Applying learning rate decay or cycling may also help to optimize and generalize our neural networks and lead to better models. This could lead to more gradual and continuous learning in later epochs. Using transformers could lead to more optimal models with faster training as well. [10]

Varying types of ensembling could also offer an improvement in the results. Other ensembling methods leveraging greater than two models, bootstrap aggregation, or applying a voting classifier as the ensembler are other potential methodologies to generate the ensembled model.

Lastly, offering lyrical data as support for this raw audio would help in any of the 'common-misclassifications' described above due to common features across different moods.

7. Work Division

Refer to Table 5 for a breakdown of the work done by each team member.

Student Name	Contributed Aspects	Details
Julia Contarino	Data Gathering and RNN Implementation	Found playlists/songs for each mood for this project and implemented the RNN model. Tuned the RNN model to improve results.
Blake Heimann	Implementation and Analysis	Trained the LSTM of the encoder and analyzed effect of various hyper-parameters. Feature engineered raw audio into dataloader ready arrays.
Anjana Shaji	Audio Features Gathering and Implementation	Scraped the audio features for this project and trained the MLP, CNN models. Tuned hyper-parameters to improve results.
Sara Srivastav	Implementation and Analysis	Data Preprocessing and initial implementation of the Multilayer Perceptron. Worked on the Ensemble model and its hyper-parameter tuning

Table 5. Contributions of team members.

References

[1] Nagesh Singh Chauhan. Audio data analysis using deep learning with python, 2020. https://www.kdnuggets.com/2020/02/audio-data-analysis-deep-learning-python-part-1.html. 2

[2] Mingwen Dong. Convolutional neural network achieves human-level accuracy in music genre classification. 2018. https://arxiv.org/abs/1802.0. 2

[3] Delbouys et al. Music mood detection based on audio and lyrics with deep neural net. 2018. https://arxiv.org/abs/1809.07276. 1

[4] Hu et al. A framework for evaluating multimodal music mood classification. 2016. https://asistdl.onlinelibrary.wiley.com/doi/10.1002/asi.23649. 1

[5] Huang et al. Music genre classification, 2018. http://cs229.stanford.edu/proj2018/report/21.pdf. 1

[6] Laurier et al. Multimodal music mood classification using audio and lyrics. 2008. https://ieeexplore.ieee.org/document/4725050. 1

[7] Liu et al. Cnn based music emotion classification. 2017. https://arxiv.org/pdf/1704.05665.pdf. 2

[8] Palanisamy et al. Rethinking cnn models for audio classification. 2007. https://arxiv.org/pdf/2007.11154v2.pdf. 2

[9] Purwins et al. Deep learning for audio signal processing. 2019. https://arxiv.org/abs/1905.00078. 2

[10] Verma et al. Audio transformers: Transformer architectures for large scale audio understanding. 2021. https://arxiv.org/pdf/2105.00335v1.pdf. 6

[11] Zain Nasrullah and Yue Zhao. Music artist classification with convolutional recurrent neural networks. 2019. https://arxiv.org/pdf/1901.04555.pdf. 2

[12] Michael Nuzzulo. Music mood classification. 2015. https://sites.tufts.edu/eeseniordesignhandbook/2015/music-mood-classification/. 1

[13] Jingwen Zhang. Music feature extraction and classification algorithm based on deep learning. 2021. https://downloads.hindawi.com/journals/sp/2021/1651560.pdf. 2