An Exploration of Deep Learning for Processing Musical Sounds

Dominic Chang *

October 13, 2022

1 Abstract

Here, we utilized 3 different models, MLP, MLP-LSTM, and CNn-LSTM to solve music genre and emotion classification problems. After initial challenges with the emotion classification dataset due to unlearnable features, we were able to utilize the MLP with the genre dataset and acheive 50% accuracy with the model.

2 Introduction

Music is a big part of society and much research has been done on different aspects of music. Here, we seek to examine music features and be able to make inferences, whether associated emotions or genre. For our project, we found two different datasets for our experiments. One was from a Kaggle Challenge [1] and the other was from a University of California San Diego Computer Audition Lab 500 (CAL500) dataset [3].

3 Models and Learning

We used three different models for our two different tasks: a MLP model for genre classification and MLP-LSTM and CNN-LSTM models for emotion classification.

3.1 The Multi Layer Perceptron

The Multi Layered Perceptron (MLP) is a Neural Network model comprised of neurons called perceptrons that is often used for continuous functions, meaning it works great with our multi-class classification task. It contains 3 layers: the input layer, which takes in all of our input features, the hidden layer, which receives information from the input layer, and the output layer, which has one perceptron that outputs a certain prediction. In each layer, a certain perceptron receives a certain amount of input features from the previous layer and then computes a weighted sum, comprised of all the input features with the associated weights, which determine how much influence certain features hold. The sum is then passed into an activation function, which is used to learn non-linear transformations and truly make the model multi-layered. We use ReLU, or Rectified Linear Unit, which takes in an input and ouputs the value if it greater than 0 and 0 otherwise [2].

3.2 The Long Short Term Memory Model

The Long Short Term Memory (LSTM) model is another type of Neural Network that builds upon the Recurrent Neural Network (RNN). The RNN is similar to a classic feed forward neural network except for the fact that it includes additional connections that analyzes sequences of inputs, usually through time. The LSTM adds onto RNNs with longer memory and forget gates that determine what continues on through time.

However, we wanted to process our data before feeding it into the LSTM. For our two experiments with the CAL500 experiment, we utilized an MLP and CNN as preprocessing before the LSTM.

*Advised by: Han Zhao, Assistant Professor, Computer Science, University of Illinois Urbana-Champaign

For the MLP-LSTM model, we fed our data into the MLP, which then outputted features to then feed into the LSTM. However, it was more complicated with the CNN-LSTM model. We kept the MFCC and Mel features separate and passed both into Convolutional Neural Networks, which are similar to MLPs but use sliding kernels and work faster. Essentially, we can choose kernels with different heights, widths, strides, and paddings. We chose to have the kernel to have one fourth of each of the dimensions of the feature arrays and the strides to have one half of each kernel dimension. Not only this, but we also have an output of five channels as to improve results. We flatten this channel dimension into the feature dimension, then concatenate the two feature arrays. That then gets passed into the LSTM. The output of the LSTM in both models is passed into a max layer, then another MLP.

4 Experiments

After creating all three types of models, we proceeded to test them on the respective datasets. We used accuracy as our main test.

4.1 Testing the Multi Layer Perceptron

Our first dataset from the Kaggle challenge had provided inputs of song metadata features like danceability, energy, key, etc., as well as other unuseful tags like popularity or song name, which were removed. The outputs were just numbers ranging from 0-10 denoting the labeled genre of the song, which meant that for this task a simple MLP, Multi-Layered Perceptron would work best.

The data set's different numerical features like acousticness are all of different ranges. After we removed all the features unrelated to the actual song features, we normalized all of the other data. This involved finding the mean and standard deviation of each category and standardizing each data value by subtracting the mean, then dividing by the standard deviation. After that, we created an MLP with the same input shape of the resulting feature array and the output shape of the number of genre classes to choose from.

For our model optimizer, we used Adam. Our loss function was Cross Entropy Loss because it works best with classification tasks like ours.

4.2 Testing both LSTMs

Our second dataset from CAL500 provided Mel-Spectrogram and Mel-Frequency Cepstral Coefficients (MFCC) sequential inputs, or inputs with a time component. The outputs were a list of different labels for each song, which ranged from instruments used to associated emotions. However, we only require emotions, which meant sorting out all other unnecessary tags after combining all of each song's features into an array. Then, for the MLP-LSTM, we combined the MFCC and Mel features along the time dimension in order to have one input. Lastly, in order for the model to learn correctly, we transformed the outputs to become mask and true label arrays. This was because our emotion list included ”not” or ”un” emotions like ”unhappy” or ”not angry” that we wanted to combine with their counterparts. The mask array included 1s for every labeled emotion, where if ”not” or ”un” emotions were labeled, their counterpart would be marked 1. Then, the true label array had 1s for every labeled emotion that was not ”not” or ”un.” This meant creating arrays of the same length as number of emotions and checking for each emotion and labeleing both arrays. These output arrays and input MEL and MFCC features were then passed into the model.

Since we are using LSTMs, we use BCEWithLogitsLoss as our loss function. Again, we use Adam for our optimizer.

4.3 Results

After tuning our models we were able to obtain results from all three of our models. Our MLP model was able to obtain around 50% accuracy on the testing set. On the other hand, our LSTM models did not seem to train on the emotion dataset. The recorded accuracies for each emotion stayed the same through many epochs of training, showing that the model was guessing the same emotions every single time.

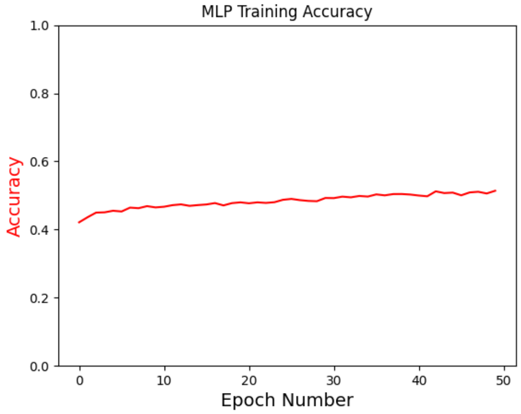

Figure 1: Here we measured the performance of our MLP model as it trained through 50 epochs. We used accuracy as a measure of how well the model was doing.

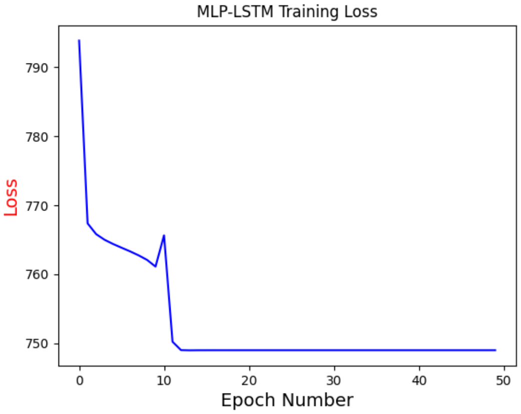

Figure 2: We measured the performance of our MLP-LSTM model as it trained through 50 epochs. We used loss as a measure of how well the model was doing.

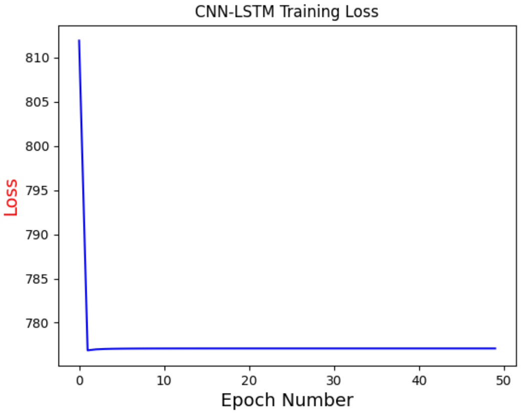

Figure 3: We measured the performance of our CNN-LSTM model as it trained through 50 epochs. We used loss as a measure of how well the model was doing.

5 Conclusions

Overall, we were able to build functioning models and acheive 50% recognition with one of our models. As a next step, we are looking to improve our models, preprocessing methods, and datasets.

References

[1] Purushottam Malgi. Music genre classification, Aug 2021.

[2] Tim Menzies, Ekrem Kocagüneli, Leandro Minku, Fayola Peters, and Burak Turhan. Chapter 24 - using goals in model-based reasoning. In Tim Menzies, Ekrem Kocagüneli, Leandro Minku, Fayola Peters, and Burak Turhan, editors, *Sharing Data and Models in Software Engineering*, pages 321–353. Morgan Kaufmann, Boston, 2015.

[3] Douglas Turnbull, Luke Barrington, David Torres, and Gert Lanckriet. Semantic annotation and retrieval of music and sound effects. *IEEE Transactions on Audio, Speech and Language Processing*, 16(2):467–476, February 2008.