CS615 Final: Music Genre Classification using Deep Learning

Anupama Achuri, Hannah Wurzel, Mai Tran, Mahir Pirmohammed

I. Abstract

Music genres are often manually classified by the instrumental techniques used throughout a song. Due to the manual effort of this task, mistakes are bound to occur in classification. Correctly classifying songs into different genres is a vital task in many different ways. For instance, Spotify uses music genres to curate individualized playlists for their users. Radio stations such as iHeartCountry rely on valid music genre classifications to play solely country music. The validity of these genres is what make users enjoy listening to their playlists or favorite radio station.

In our study, we attempted to classify music into ten different genres using deep learning fundamentals. We used two different deep learning frameworks: a convolutional neural network (CNN) and an artificial neural network (ANN). In this paper we will demonstrate how our deep learning network was able to classify music by its genre.

II. Background/Related Work

There have been an abundance of studies on music genre classification. Many of these studies have used deep learning architectures such as a CNN to accurately classify music. K M et al (2021)[2] used a CNN model that used an ADAM optimizer to classify music genres. There architecture resulted in a 94% accuracy rate with all but one of the genres being classified at a 90% rate or higher. This great success gives us hope that our CNN we are building from scratch will be successful.

One of our first steps in the process of creating a music genre classification model was figuring out what our architecture should look like. Choi et al (2017)[0] described three different CNN architectures that are suitable for music genre classification. Their best performing model was a CNN that consisted of five convolutional layers, 2 fully-connected layers and two sigmoid activation layers. This model used a kernel with size two.

The data set that will be used in this study contains a variety of musical features. Descriptors like spectral centroid and spectral bandwidth were terms that we as a group were unfamiliar with. We turned to Tzanetakis et al (2002)[1] to learn more about the different features contained in a song. The goal of that paper was to explore how different features play a role in music genre classification. Using only three features, timbral texture, rhythmic content and pitch content, Tzanetakis et al were able to classify music genres at a 61% accuracy.

III. Data

The data comes from a popular data set called GTZAN. This data set is often used for music genre classification problems similar to what we are solving. We retrieved it from Kaggle.

This data set consists of four components. The first component is 100 thirty-second audio files for each genre of music. There are 10 genres total in this data set: rock, classical, metal, disco, blues, reggae, country, hip-hop, jazz and pop. So, all together there are 1000 audio files. The second component contains the Mel Spectrogram images for each audio clip. A Mel Spectrogram depicts the waveforms of an audio clip. An example of this can be seen below in Figure 1. The next component is a CSV file containing the features of the 1000 audio clips. The final component is another CSV file of audio features except this one breaks the thirty-second audio clips into three-second chunks. This allows there to be ten-times the amount of data.

For our implementation we will be using the second component: the images of the Mel Spectrograms.

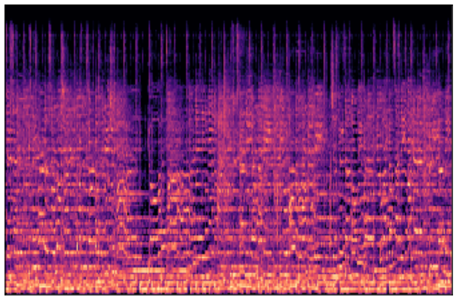


Fig. 1: A rock Mel Spectrogram

IV. Architectures and Hyper-parameters

We will be classifying songs based on their genre using two separate architectures: a convolutional neural network and an artificial neural network. We thought this dataset would be perfect for a convolutional neural network (CNN) because of the Mel Spectrogram images that are provided. We hoped that images for each genre would have distinct enough qualities for our models to correctly differentiate between them. Additionally, we chose to use a artificial neural network (ANN) as well because of the success seen in other multi-class classification problems.

A. *Convolutional Neural Network*

Our CNN uses mini-batches to learn. It is comprised of an Input image layer (70 x 218 x 336), a convolutional layer with

Kernel size (3 x 3), a ReLu layer (70 x 216 x 334), a Max pool layer with pool size (2 x 2), a Drop out Layer (p = 0.2), another convolutional layer with Kernel size (3 x 3), a ReLu layer (70 x 106 x 165), a Max pool layer with pool size (2 x 2), a Drop out Layer (p = 0.2), a flattening layer (70 x 4346), a Drop out layer (p = 0.3), a fully connected layer (4346 x 10), a Softmax activation layer (70 x 10), and finally a cross-entropy layer (70 x 10).

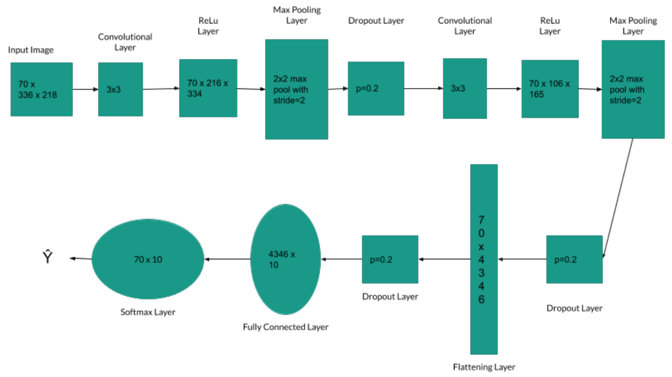


Fig. 2: Our CNN Architecture

First, we processed the data with colors but then realized we only need the brightness of a line to extract features for the convolutional layer so the input data was gray-scaled.

Then we ran the architecture with 100 epochs and a learning rate of 0.001 for all convolutional and FC layers. We also use ADAM Learning for the FC layer, which means Adaptive Moment and is a popular optimization algorithm for deep learning that combines the benefits of both momentum and adaptive learning rate methods.

This is the best result that we can get with CNN. Even though training accuracy is only 45.57% and validation accuracy: 32.33%, the model is not overfitting. Other models will be discussed in the later part.

B. Artificial Neural Network

Our ANN is comprised of an input layer (700 x 73249), a dropout layer (p = 0.2), a fully connected layer (73249 x 800), a ReLu activation function (700 x 800), another dropout layer (p = 0.2), another fully connected layer (800 x 10), a Softmax activation function (700 x 10) and finally a cross-entropy layer (700 x 10). We added drop out layers before each fully connected layer to improve the generalization of the model.

The input data was gray-scaled since we want to learn on the different bright line values of the Mel Spectrogram for the different genres. This was also Z-scored for ANN through our Input Layer.

For our hyper-parameters, we also went with 100 epochs and a learning rate of 0.001 for ADAM learning of FC Layers.

We also tested out multiple different sizes for our fully connected layers. However, due to limited resources, we were only able to get to a max of 1500. However, our fully connected layer with size 850 produced the best results.

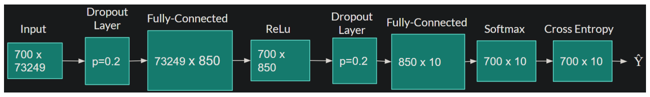


Fig. 3: Our ANN Architecture

V. Evaluation

A. CNN

All the CNN architectures were only able to learn around 50% maximum. That is because we could not extract many features due to a limited number of filters, 1 for each convolutional layer.

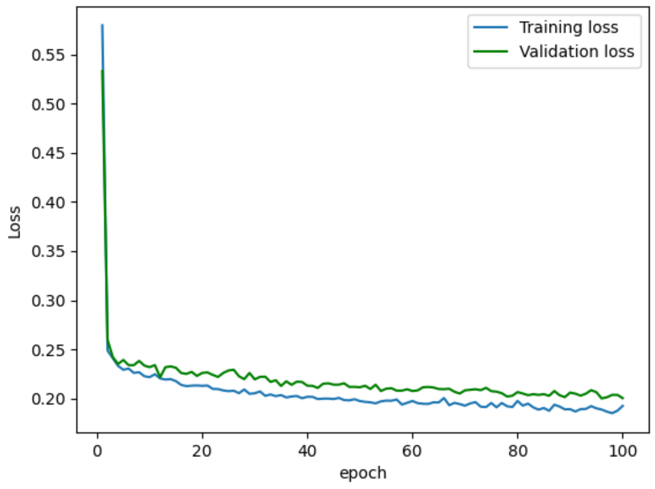
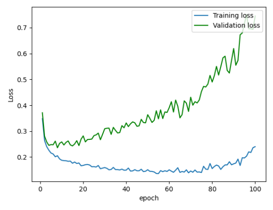
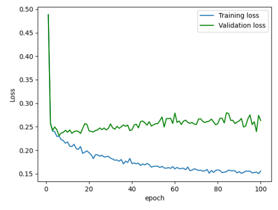


Fig. 4: CNN best result

Putting a dropout layer after each pooling layer or before the FC layer definitely makes the model less over-fit. Figure 5 is the result from keeping only (a) 1 drop out layer before the FC layer and (b) no drop out layer. We can see more over-fitness compare to the best result above.



(a) With a drop out layers



(b) With no drop out layers

Fig. 5: CNN results

Architecture	Training Accuracy (%)	Validation Accuracy (%)
2 convolutional layers, no ReLu layers, 1 drop out at FC Layer	33.8571	26.3333
2 convolutional layers, with Relu layers, 1 drop out at FC Layer	70.4285	30.0
2 convolutional layers, with Relu layers, 3 drop out layers	45.5714	32.3333
2 convolutional layers, with Relu layers, no drop out layers	50.8571	32.6666
3 convolutional layers, with Relu layers, no drop out layers	53.8571	21.6666

TABLE I: Comparison of accuracy's for different CNN architectures

In Table 1 we compare all the architectures we have tried. Even though some models have better training accuracy, their generalization is bad.

B. ANN

We had 10 genres with 100 Mel Spectrograms so 1000 images in total to learn on. Through our ANN architecture we were able to learn really well on the training data as seen in Figure 3. However, our model did not generalize well for our validation set. We were able to obtain a Training accuracy of 97.1429% but a validation training of 34.4482% in our best model.

In order to increase the generalization, we implemented dropout layers but this did not significantly improve our results as can be seen in the Table 1. To reinforce the hyperparameters discussed earlier, we also tested not using ADAM learning and different first FC Layer Output Sizes.

Architecture	Training Accuracy (%)	Validation Accuracy (%)
No ADAM, 500 epochs, First FC Layer Output Size of 10000	22.1428	19.3979
ADAM - First FC Layer Output Size of 1500	95.9302	32.4415
ADAM - First FC Layer Output Size of 850	97.1429	34.4482
ADAM - First FC Layer Output Size of 800	96.8429	34.1082
ADAM - No Dropout Layers - 800 epochs	83.7142	33.7793

TABLE II: Comparison of accuracies for different ANN architectures

VI. Conclusions / Analysis

A. CNN

Our CNN model's couldn't extract many features and doesn't learn much over each epoch. However, compared to other architectures, this one is the best because it doesn't overfit and still learns all the way to 100 epochs with good generalization.

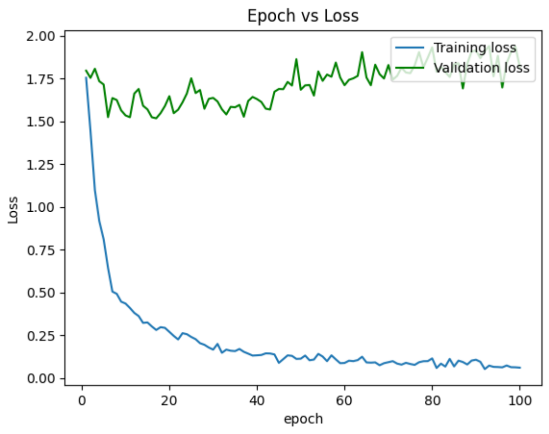


Fig. 6: ANN Results: Epoch vs Loss Plot for Training and Validation

B. ANN

Due to the nature of the input data being a large flattened image of 73249 features and a time series, the ANN architecture was too small to be able to accommodate the many features and generalize for the validation data. Because we were using ADAM learning to increase the training time, this caused memory overload issues and therefore we could not increase the output layer size of our first fully connected layers to extract more of the 733249 features.

VII. Future Work / Extensions

A. CNN

As mentioned in the conclusion, we need to implement more filters for each convolutional layer, like those ML frameworks. Therefore, we can learn at different parts of the image, gather more features, and be able to improve the current model.

We can also try to implement an average pool layer, to compare with the result of the current max pool layer.

Last but not least, we can try to implement RNN and CRNN to compare with the result CNN. Actually, in one of the papers that we based the work on, they mentioned CRNN outperforms all other models.

B. ANN

Based on our findings we believe it could be super useful to run our architectures on machines with larger Memory/GPU capabilities. Our personal computers do not have enough computing capabilities to be able to run such a large data set. We believe if we had the resources, our models could perform much better.

VIII. Bibliography

[0] Choi, Keunwoo, et al. "Convolutional Recurrent Neural Networks for Music Classification." 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017, https://doi.org/10.1109/icassp.2017.7952585.

[1] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," in IEEE Transactions on Speech and Audio

Processing, vol. 10, no. 5, pp. 293-302, July 2002, doi: 10.1109/TSA.2002.800560.

[2] K M, Athulya and S, Sindhu, Deep Learning Based Music Genre Classification Using Spectrogram (July 10, 2021). Proceedings of the International Conference on IoT Based Control Networks Intelligent Systems - ICICNIS 2021, Available at SSRN: https://ssrn.com/abstract=3883911 or http://dx.doi.org/10.2139/ssrn.3883911