2018

Genre Recognition: Keras Audio Classification

BIG DATA TECHNOLOGY

A. BURROWS, J. JACKSON, N. ZAMORA

Classifying Music

A. Burrows, J. Jackson and N. Zamora

***Abstract*-This is an audio classification study looking at classifying eight thousand songs by genre using neural networks at three different dimensions. Overall, the outcome of this study shows the power of machine learning to classify at a better than chance rate.**

I. Introduction

Audio classification is an interesting way to use machine learning to utilize unstructured data. In this study, we used snippets of songs (30 seconds) to try to classify them by genre. These eight genres are hip-hop, pop, folk, rock, experimental, international, instrumental, and electronic. Our data set comes from the Free Music Archive (FMA). The FMA is an "an interactive library of high-quality, legal audio downloads" [1]. Reference [1] includes a link to the data source. We used the smallest set of songs including approximately eight thousand tracks and eight balanced genres. Our goal for this study is to achieve a validation score that is better than chance, which is approximately 12.5% (or one in eight) accuracy.

A. Related Work

Audio classification can be used for many things, from classifying audio files by their features to creating voice to text recognition that is now available on every smart phone. One major application of audio classification is Shazam. This application allows users to search for songs by recording a quick ten second snippet of the song. This snippet is then compared to Shazam's database to match it to the correct song and returns the name, artist and even links on where to download the song. Another widely known application of audio classification is speech synthesis; this allows artificial voice to be used in conversational agents.

II. System Model

B. Methodology

During our study, we created many different models, however we would like to highlight our three best models. Before creating our models, we first had to extract the features out of each song. We extracted a feature called Mel-frequency cepstral coefficients (MFCC) using the librosa package. The MFCC mimics the logarithm perception of loudness pitch of the human auditory system and tries to eliminate speaker dependent characteristics by excluding the fundamental frequency and their harmonics. Once we extracted the MFCC from every song, we created all three models using Keras, which is a package in Python that runs on top of Tensorflow to create neural networks. All three models are using a seventy-thirty split of the data. This means the model is trained on seventy percent of our data and then validated on the remaining thirty percent. Prior to being modeled, all our data has been scaled using StandardScaler from sklearn, which standardizes by removing the mean and scaling to the unit variance.

Our first model was a deep neural network (DNN). For this model, we used the mean of the MFCC for 80 features as our testing parameter. We will refer to this model as 2D. The process for which we created our 2D model is as follows: create a dense layer activated using the relu function and dropout fifty percent to avoid over fitting. This is

repeated for two layers followed by an output layer activated using the softmax function. Our 2D model includes one visible layer and one hidden layer. There are 88,584 trainable parameters in this model and was run for 100 epochs, which took approximately three minutes to completion.

Our second model was a convolutional neural network (CNN) using convolutional one-dimensional layers. The process for creating a CNN is like a DNN, but in the end of creating the model you must flatten the model back to one-dimensional to create the output nodes for the eight genres. For this model, we used the full extraction of 80 features each with 2500 values (not averaged) as our testing parameter. We will refer to this model as 3D. Our 3D model includes two visible layers each structured as follows: convolutional layer, pooling layer, batch normalization, and a fifteen percent dropout. The convolutional layers contained 32 and 64 filters, respectively, while utilizing an l2 kernel regularizer. There are 255,656 trainable parameters in this model and was run for 50 epochs, which took approximately twenty-five minutes to completion.

Our third and final model was also a CNN using convolutional two-dimensional layers. For this model, we used 96 melgrams with 1366 values, which was computed by multiplying the duration of the song by the sampling rate, for each song snippet, with an additional axis added to allow for four dimensions as our testing parameter. We will refer to this model as 4D. Our 4D model includes four visible layers structured in the same pattern as the 3D model. The convolutional layers contained 32, 64, 64, and 32 filters, respectively. There are 2,796,648 trainable parameters in this model and was run for 50 epochs, which took over seven hours to completion.

C. Results

Our 2D model was overall a success as it classified better than random chance. In Figure 1, you can see the results for our 2D model, this model was trained at a 69.1% accuracy and then validated to a 48.3% accuracy for classifying into the correct genre. Figure 2 shows the loss of our 2D model; notice that the validation loss increases across the epochs. This illustrates the need for fine-tuning parameters and possibly adding more layers. Lastly, in Figure 3 you can see the confusion matrix of this model. We know this is predicting well as the diagonal is the darkest part of the matrix, indicating the true positive rates. Even though the model appears subpar, most of each genre is classified accurately.

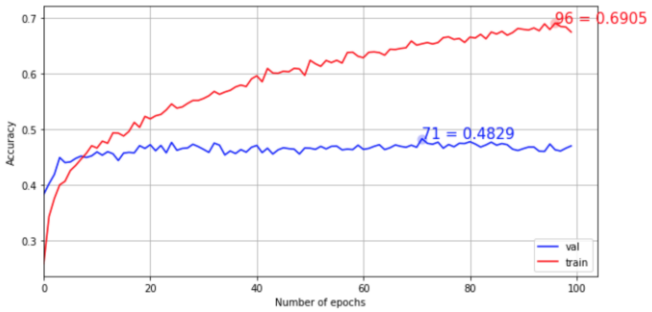

Figure 1: 2D Results

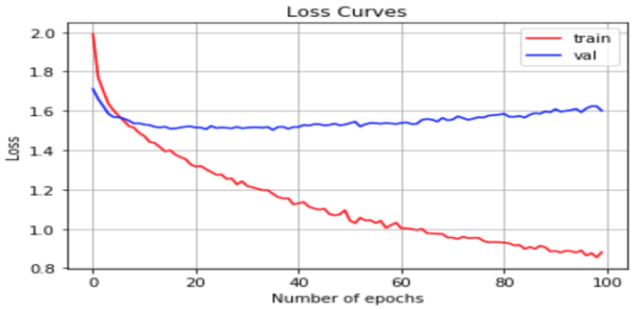

Figure 2: 2D Loss Curve

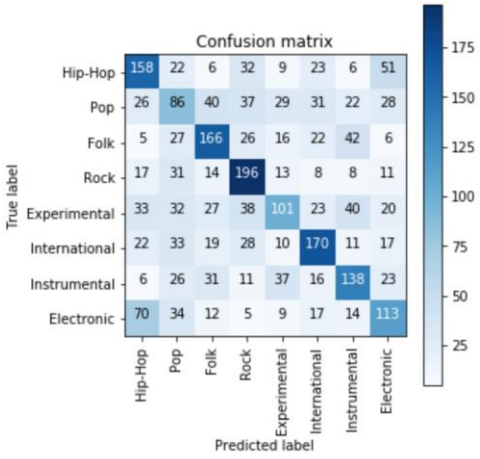

Figure 3: 2D Confusion Matrix

Our 3D model was also overall a success as it predicted accuracy better than chance. In Figure 4, you can see the results for our 3D model; this model was trained at a 52% accuracy and then validated to a 44% accuracy classifying into the correct genre. Figure 5 shows the loss of our 3D model; notice that the training and validation loss curves decrease in perfect harmony. This is indicative of a decent-performing model that could be continuously improved by adding more layers, filters, or testing various optimizers. Lastly, in Figure 6 you can see the confusion matrix of this model. This model also has the darkest values along the diagonal, which describe the true positive music genres.

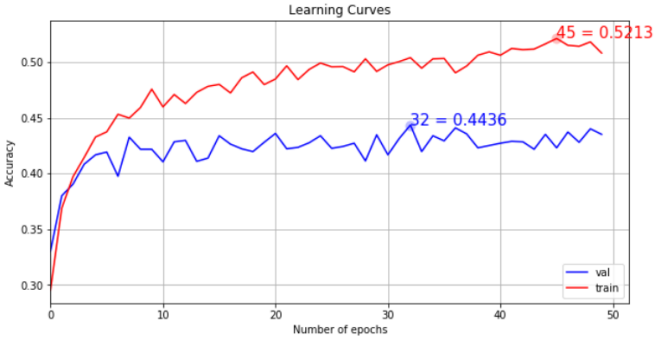

Figure 4: 3D Results

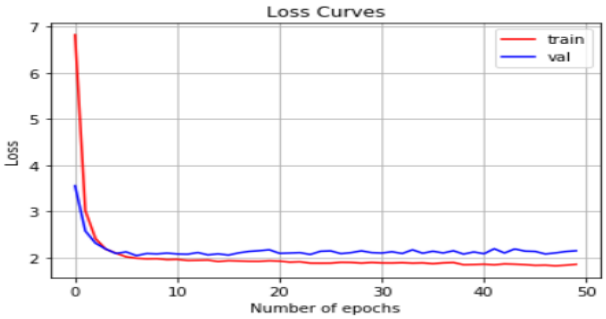

Figure 5: 3D Loss Curve

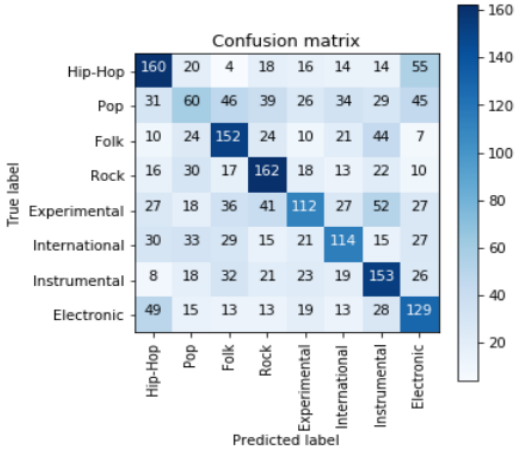

Figure 6: 3D Confusion Matrix

Lastly, our 4D model was also a success as it predicted at an accuracy rate better than 12.5%. In Figure 7, you can see the results for our 4D model; this model was trained at a 96.8% accuracy and then validated to a 42% accuracy for classifying into the correct genre. Clearly this model needs serious work due to the obvious overfitting of the training data. Figure 8 shows the loss of our 4D model; notice the terrible zig-zag pattern harshly increasing across the epochs. This describes a poorly-performing model that needs plenty of fine-tuning on layers, filters, parameters, optimizers, and regularizers. Lastly, in Figure 9 you can see the confusion matrix of this model. Even though the loss curve in Figure 8 displays a below favorable return, the confusion matrix reports decent accuracy following the same trend as the previous models.

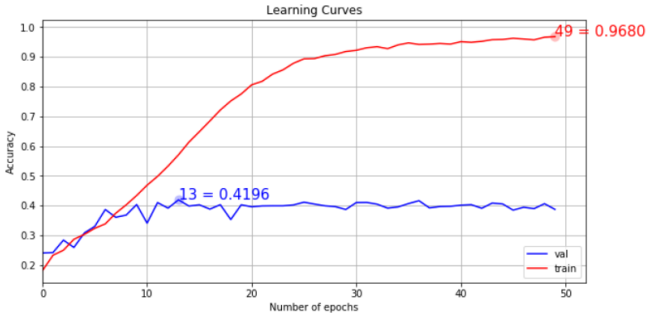

Figure 7: 4D Results

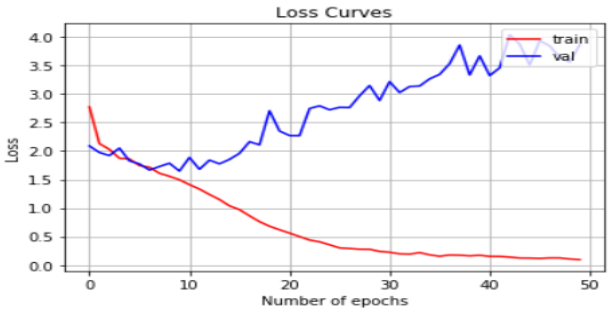

Figure 8: 4D Loss Curve

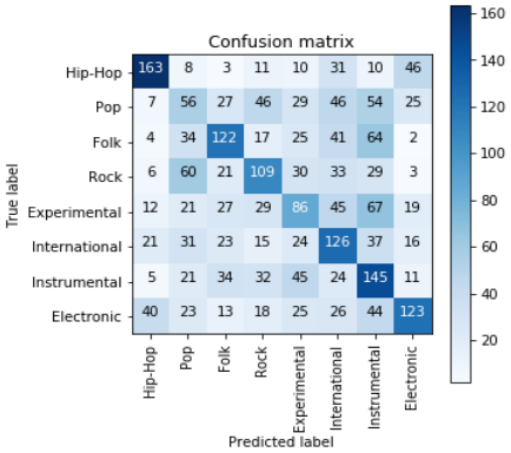

Figure 9: 4D Confusion Matrix

III. CONCLUSION

Overall, we are happy with our results as all three models were better than chance at classifying songs by genre. We are worried our training may have overfit the data due to our layers and parameters used in the CNN and the lack of power in big data multi-classification for the DNN. The 3D model performed best overall because there was minimal, if any, overfitting on the training data and the loss functions were in downward harmony. While the testing accuracy was not highest for the 3D model, the results clearly indicate the foundation is strongest and the room for improvement is available from the current structure. Both the 2D and 4D models would require rebuilding from the beginning to optimize the results.

D. Future Work

One thing we would like to try in the future is segmenting the features into blocks of time, for example, breaking the 30 seconds into 6 partitions of 5 seconds each with a 10ms overlay, to better represent the feature. Another thing we would like to see in the future is a graphics-processing unit (GPU), which would be helpful for testing 4D models because it gets computationally exhausting at that level of dimensions. And finally, there are

different features that can be extracted using the librosa package, so we would like to see if using a feature besides MFCC would better classify our songs into genres.

ACKNOWLEDGMENT

A. J. N. would like to thank Professor Paul Rad for encouraging us to step outside our comfort zones and pushing us to work on something new, as well as for all his guidance throughout this project. Special thanks to Arun Das for providing his personal melgram function to help with our 4D analysis, and also for providing his advice and unique perspective on the project.

REFERENCES

[1] "Mdeff/Fma." *GitHub*, github.com/mdeff/fma.
[2] Python lab link: http://129.114.108.202:8888/lab?