Music Genre Classification Using Deep Learning

Matthew Meza[*]
MS Candidate in Electrical Engineering, Stanford University

Job Nalianya[†]
Coterminal Degree candidate, Electrical Engineering, Stanford University
(Dated: March 23, 2018)

We apply deep learning to the problem of music genre classification. In this project, we design and train a Convolutional Neural Network (CNN) capable of classifying 128x128 pixels spectrograms of music into 8 genres with 92.7% accuracy. This CNN is then used to classify spectrogram slices obtained from a single song and the resulting ensemble average used to classify the song into a genre with high accuracy. Our approach is advantageous as it allows music genre classification without the need for extensive prior knowledge of music and audio features that were traditionally used for this application.

I. INTRODUCTION

The rapid growth of the digital entertainment industry in the recent years makes automatic music genre classification desirable. Correct music genre classification can play a vital role in such businesses as listeners are likely to pick a song over another based on the song's genre as opposed to the song's contents. Furthermore, consumers are likely to browse music by genre as compared to by other meta-data such as artist [1]. The importance of solving this problem is further compounded by the continued interest in Music Information Retrieval (MIR) - a field concerned with accessing, analyzing and organizing large collections of music.

In this project, we apply deep learning to the task of Music Genre Classification. We use music audio spectrograms as the input to our network which is a mid-level representation approach as described in [5]. At first glance, the use of spectrograms for this task presents a deluge of data which is challenging to learn information from. One approach would be the use of a Recurrent Neural Network (RNN) to run through the spectrogram or raw audio of the entire song to classify it. This is however computationally heavy and may be difficult for the network to learn the necessary general attributes to determine a genre.

Instead, we use an image representation approach. We obtain the spectogram of a song of choice and split the spectrogram into 128x128 pixel slices that represent about 2.56s of audio. We then train a CNN to be able to classify each of the slices into one of 8 music genres shown in table I. The trained CNN is then used to classify spectrogram slices from a single song whose genre needs to be determined. The resulting genre classifications of the different slices are then used to vote and determine the most likely genre of the song in question. This approach allows us to view the task as an image classification problem and hence take advantage of techniques used in computer vision tasks. Moreover, this approach improves the accuracy of the classifications owing to the voting between the different spectrogram slices.

II. RELATED WORK

Several approaches have been used in addressing this problem. Most traditional approaches involve the use of hand-crafted features such as the Mel-frequency cepstral coefficients (MFCCs) as an input to a network; see [10]. These approaches are disadvantageous as they heavily rely on the prior specific knowledge of the ML experts training the network to handcraft the features. This makes it difficult to scale [2].

With time, there has been increased preference to learning of the features as opposed to hand-crafting them. This is advantageous as the network is able to learn features on its own which is potentially more efficient and effective as opposed to being constrained to features known to the ML experts. This allows for scalability and frees up the ML experts time to work on high level problems such as hyper-parameter tuning. For instance [3] uses Convolutional Deep Belief Networks (CDBNs) to learn features and classify music into five genres with classification accuracy equaling or surpassing that obtained with MFCCs. [4] on the other hand trains a CNN on beat-aligned timbre and chroma features obtained using an unsupervised pretrained network on the Million Song Dataset. These approaches however depend on the availability of a large amount of data.

More recent approaches, like the one we use in this paper, make use of the visual representations of audio signals in the form of spectrograms. The spectrograms are then used as an input to a CNN enabling the use of image classification techniques used in computer vision. For example [2] use MFCC spectrograms as an input to a CNN to classify music.

Other approaches that do not solely depend on audio features have been proposed. For instance, [6] proposes the use of a combination of audio, text and images to

* mattmeza@stanford.edu
† jobn@stanford.edu

perform multi-label music classification. [7] on the other hand analyses customers reviews and uses them to classify the music. These two approaches are particularly insightful as they promise to "add context" to the music genre classification task. Moreover, the multi-label classification approach addresses the fact that most songs belong to different genres. We did not however take this approach given the scope of our project.

Yet another approach that aims at reducing the dependency on prior knowledge and human defined features involves the use of end-to-end deep learning as discussed in [5]. This involves the use of raw audio as the input to a classifier - no need to extract features such as MFCCs or spectrograms. While the approach presented in [5] did not outperform the use of spectrograms in terms of classification accuracy, they showed that end-to-end deep learning on raw audio is capable of autonomously discovering frequency as well as phase features in audio.

III. DATASET AND FEATURES

A. Dataset Description

We obtain our dataset from the Free Music Archive (FMA) that is available online at `http://freemusicarchive.org/`. The archive provides more than one-hundred thousand tracks under the Creative Commons Licence to help alleviate the data scarcity problem and facilitate MIR. The description of the dataset can be found in [8].

We downloaded 8000 30s long tracks from the archive at `https://os.unil.cloud.switch.ch/fma/fma_small.zip` in the *fma_small* dataset. This dataset is summarized in table I below.

Label	Genre	Number of Tracks
0	Electronic	1000
1	Experimental	1000
2	Folk	1000
3	Hip-Hop	1000
4	Instrumental	1000
5	International	1000
6	Pop	1000
7	Rock	1000

TABLE I. *Summary of the fma_small dataset*

Given the format of this dataset and the fact that the tracks are not already labelled in the dataset, we spent some time pre-processing and cleaning up the dataset before training. Section III B describes the pre-processing steps and scripts we wrote to aid the process. The scripts and model can be found in our code repository at `https://github.com/jonalkn/cs230_final_project`

B. Data Preprocessing

The first step in the data processing pipeline involves labelling the tracks with their genres. As a result, we wrote a script named *classify_save.py* that appends a genre label (a number from 0-7) as the first part of each track's name. This was followed by converting the .mp3 files to .wav format using *convertMP3toWAV.py* script. This was motivated by the fact that we are using the Sound eXchange (SoX) package to obtain the spectrograms of the tracks and the package only supports the .wav format. Using the SoX package, we converted the tracks to mono and obtained a gray scale spectrogram. We decided to convert the tracks to mono since stereo-information is not particularly useful for our network to learn. Additionally, we used a grayscale spectogram because it is more computationally efficient and there is no extra information in using a color spectogram. Most audio is recorded at 44,100 samples per second which would correspond to 44,100 pixels per second for a spectogram. For our purposes we decided to use 50 pixels per second of audio, see *createSpectrograms.py*, for computational efficiency and because we want the network to learn the low frequency attributes such as rhythms and syncopation.

It is worth noting that 30s of audio has an overwhelming amount of data in its spectrogram. We decided that the best approach would be to slice each track's spectrogram into approximately 2.56s long segments. This was achieved buy slicing each spectrogram into 128x128 images and using the slices as the input to our CNN. This is further advantageous as a single track provides several images with the same label further increasing the size of our dataset.

This resulted in about 80k spectrograms which were split into the train/dev/test sets using the ratio 0.9/0.05/0.05 respectively.

C. Data Augmentation

To avoid over-fitting the training data we performed data augmentation. This was achieved by adding "zoomed out" spectrogram slices to the training set. The "zoomed out" slices were obtained by taking spectrograms of all the tracks in the dataset with a 25 pixels/s as opposed to the original 50 pixels/s. This operation implies that the resulting 128x128 spectrogram represents 5.12s of audio as opposed to the initial 2.56s. This form of augmentation re-stresses the fact that we want the model to learn the low frequency attributes to an audio clip. Further information on the data augmentation is described in the experiments section.

IV. METHODS

A. Pipeline

Figure 1 shows our proposed pipeline to be used to classify a song into a genre. To start, the song audio (.mp3 or .wav) format is converted into its gray scale spectrogram. A spectrogram is a pictorial representation of the variation of the amplitudes of the different frequency components in a signal over time. Only one channel is needed therefore it is computationally advantageous to convert the stereo audio to mono. The resulting spectrogram is then split into 128x128x1 pixels slices which at 50 pixels/s represents roughly 2.56s long audio slices. The obtained slices are then forward propagated through a CNN with a softmax output giving the probability of each of the slices belonging to a given genre.

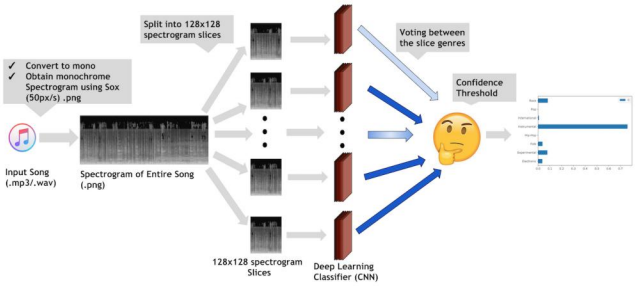

FIG. 1. *Pipeline*

The classification outputs of each slice are then averaged providing an ensemble average that is a good representation of the probability of the song in question belonging in the 8 genres. Using a confidence threshold, the song's genre can be determined. The pipeline outputs a horizontal bar graph showing the confidences for the different genres. In the following section, we describe the design of the CNN that classifies the spectrogram slices.

B. Network

1. Base Model

To get started, we experiment with a baseline CNN to classify the slices. The baseline CNN consists of 3 convolutional layers. Each layer consists of a 2D convolution, batch normalization, maximum pooling, and a Relu activation. After the 3 convolutional layers we follow with two fully connected layers with a softmax output. The performance of the CNN in classifying the spectrogram slices is summarized in table II.

From the performance of the baseline model, we conclude that the model has high bias and hence there is need to build a deeper network that fits the data better.

Set	Accuracy [%]
Train	77.1
Dev	55.6
Test	64.1

TABLE II. *Baseline CNN accuracy performance on different sets*

The design of the deeper model and its performance is discussed in sections IV B 2 and V below.

2. Model Selection Procedure

To better fit the data, we designed a deeper CNN with the architecture shown in figure 2 (Refer to the last page of this report to view an enlarged image of the CNN). Notable changes to the base network include the use of average pooling as opposed to maximum pooling for the layers; use of 5 convolutional layers instead of 3; and changing the convolutional kernel size to 7 instead of 3 for the first 4 layers and a kernel size of 5 for the last convolutional layer.The use of a larger kernel size is motivated by the relatively large image size inputs that we use.

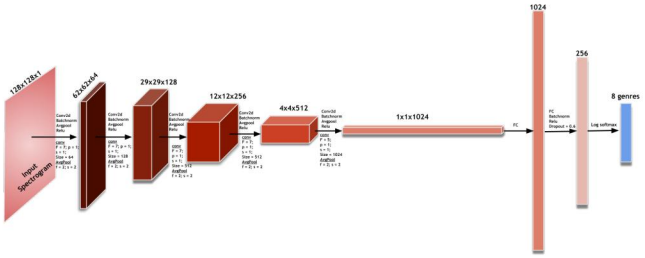

FIG. 2. *Deep Model. Refer to the last page for an enlarged version of this CNN*

We implement dropout regularization for the fully connected layers. The performance of this deep model with only dropout regularization at a rate of 0.8 is shown in table III $Model\#1$ column. It can be seen that dropout rate is not sufficient to regularize the model. We therefore ran a hyper-parameter search over dropout rates and determined that a rate of 0.6 works best. The performance of the updated model is summarized in the $Model\#2$ column in table III.

While model 2 improves the performance on the dev set, the variance is still high. We therefore decided to add L2 regularization by adding weight decay. We ran a hyper-parameter search and determined that a weight decay factor of $1e-5$ works best. Integrating this change results in the accuracy performance of $Model\#3$. At this point, we augmented the training set by adding spectrogram slices obtained at 25 pixels/s which corresponds to 5.12s of raw audio. Combining these changes and training the model results in $Model\#4$.

	Model#1 no reg. D-rate = 0.8 $lr = 0.001$	Model#2 (search dropout rates) D-rate = 0.6 $lr = 0.001$	Model#3 (search weight decay) D-rate = 0.6 $W_{decay} = 0.00001$ $lr = 0.001$	Model#4 (Add data augm.) $D - rate = 0.6$ $W_{decay} = 0.00001$ $lr = 0.001$	**Model#5** (search learning rates) Data augm. $D - rate = 0.6$ $W_{decay} = 0.00005$ $lr = 1e - 4$
Train	98	98.8	93.6	95.3	97.7
Eval	87.2	89.5	86.7	89.9	93.4
Test	87.3	89.2	86.9	89.8	92.7

TABLE III. *Accuracy for different models. From left to right, the models are updated after running experiments to choose hyper-parameters.*

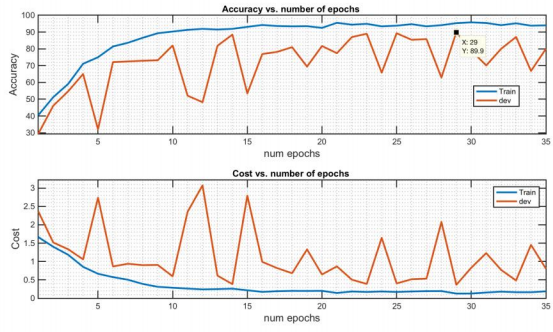

FIG. 3. *Learning curve and Cost plots for Model 4. Demonstrates the erratic nature of the dev curves with a learning rate of 0.001*

Figure 3 shows the learning curve and cost as a function of the number of epochs for model 4. It can be observed that the dev curves are very erratic. We thus assumed that this was most likely due to a large learning rate. We thus made a hyper-parameter search over learning rates and settled on a learning rate of 0.0001. We further experimented with weight decay values and chose a weight decay factor of 0.00005. Putting it all together, we picked Model 5 whose performance can be seen in the last column of table III.

V. RESULTS AND DISCUSSION

Figure 4 shows a plot of the learning and cost curves versus the training time (number of epochs) for model 5. With a smaller learning rate and increased weight decay factor, the dev curves are less erratic as compared to those in figure 3 above.

Running the test set through model 5 discussed above, figure 5 shows a plot of the resulting confusion matrix. As expected, the model does well on the test set as demonstrated by the high accuracy along the matrix diagonal. However, we can see from the confusion matrix in fig. 5 that a relatively significant portion of electronic spectrograms are being classified as Hip-Hop. The same goes for experimental genre spectrograms being classified as pop. The latter makes sense given the lose specifications of the experimental genre.

We tested the CNN in the pipeline shown in fig. 1 for classifying individual songs. For demonstration purposes, we set the confidence threshold to 60% and tested it using songs that are not from the FMA data set. One interesting result is when we classified Enya's Gladiator soundtrack. Figure 6 shows a horizontal bar graph of the classifier's confidence on the song's genre. As expected, this is a relatively hard song to classify. The model picks two top genres as Folk and Instrumental. However, none of the confidences are greater than the 60% threshold. It is fascinating to see that while the CNN was trained on single labelled spectrograms, ensembling allows it to highlight multiple genres that a particular song may belong to.

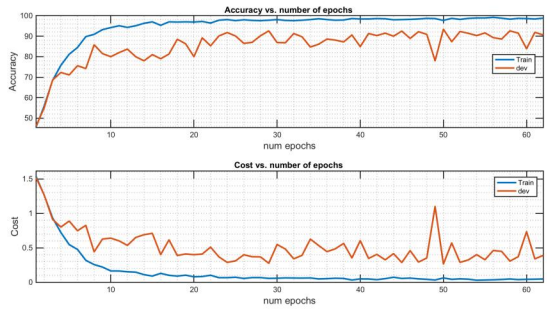

FIG. 4. *Learning and Cost function plots for Model 5. Note that with a smaller learning rate, the dev curves are less erratic as compared to the plot in figure 3*

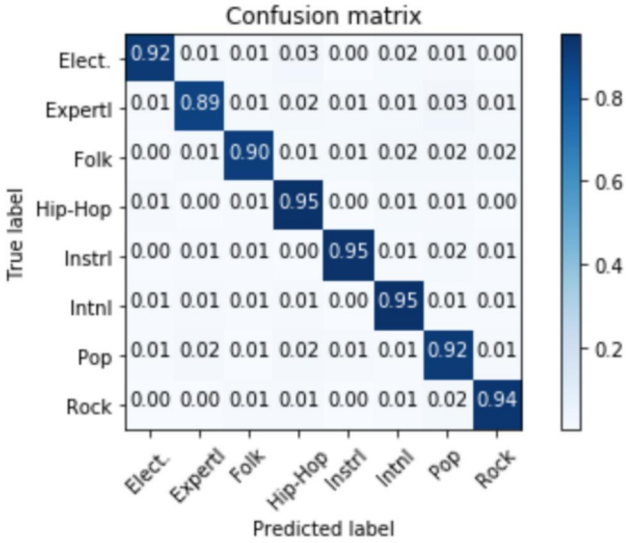

FIG. 5. *Confusion Matrix*

VI. CONCLUSION AND FUTURE WORK

In this project, we design a deep neural network and train it to classify a song into one of eight genres. The network takes in slices of the song's audio spectogram. Each slice is then categorized into one of eight genres

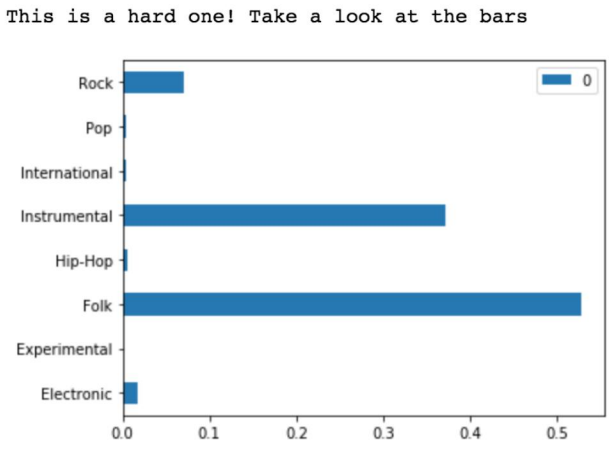

FIG. 6. *Classification output for Enya's Gladiator Soundtrack*

and an ensemble average of the classification outputs is used to determine the genre of the song. An accuracy of 92.7% is achieved when classifying individual slices in the test-set with a variance of 5% from that attained on the training set. To reduce this variance, we could increase regularization by experimenting with larger weight decay values and add more training data from the FMA. Alternatively, we would consider modifying the cost function of our current model to penalize mis-classifications (see confusion table in figure 5). For instance, penalize classification of spectrograms in the experimental genre more if they are classified as pop.

While were able to build a model capable of classifying songs into 8 genres with high accuracy, it is important to note that there exists hundreds of music genres that are not necessarily mutually exclusive. In fact, the separation of songs into different genres is very fluid, varies with time, region and culture. This is perhaps exemplified by the 'International' and 'Experimental' genres that may have very different interpretations depending on region. As a result, only assigning a single label to a song does not suffice. A future extension of the project would be implementing multi-genre classification of music that allows multi-labelling of input spectrograms.

VII. CONTRIBUTIONS

We worked together in designing the CNN model and experiments to test it out; writing the report and poster.

Job was responsible for 'data collection' and Python scripting for data pre-processing and data set preparation. He further ran the tests on the EC2 AMIs.

Matthew was responsible for the creation of the neural network architecture and implementing it using PyTorch.

VIII. ACKNOWLEDGEMENT

We would like to thank Suraj and Zahra our TAs for their suggestions and advice. We also appreciate Amazon for providing AWS credits which provided computing resources and space to train and evaluate our model.

IX. APPENDIX

The code repository for this project can be found at: https://github.com/jonalkn/cs230_final_project

[1] J. H. Lee and J. S. Downie, Survey of music information needs, uses, and seeking behaviours: Preliminary findings. in ISMIR, vol. 2004, 2004, p. 5th.

[2] T. Li, A. B Chan, and A. Chun, Automatic musical pattern feature extraction using convolutional neural network, in Proc. Int. Conf. Data Mining and Applications, 2010.

[3] H. Lee, P. Pham, Y. Largman, and A. Ng, Unsupervised Feature Learning for Audio Classification Using Convolutional Deep Belief Networks, in Advances in Neural Information Processing Systems 22. 2009.

[4] S. Dieleman, P. Brakel, and B. Schrauwen, Audio-based Music Classification with a Pretrained Convolutional Network, in Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR), 2011

[5] S. Dielman, B. Schrauwen, "End to End Deep Learning for Music Audio," IEEE International conference on Acoustic, Speech and Signal Processing, 2014

[6] S. Oramas, O. Nieto, F. Barbieri, X. Serra, "Multi-label Music Genre Classification from Audio, Text and Images Using Deep Features", https://arxiv.org/abs/1707.04916

[7] S. Oramas, L. Espinosa-Anke, A. Lawlor, et al. "Exploring customer reviews for music genre classification and evolutionary studies," In ISMIR, 2016

[8] M. Defferrad, k. Benzi, P. Vandergheynst, X. Bresson, "FMA: A Dataset for Music Analysis", online: https://arxiv.org/abs/1612.01840

[9] Li T.L.H., Chan A.B. (2011) Genre Classification and the Invariance of MFCC Features to Key and Tempo. In: Lee KT., Tsai WH., Liao HY.M., Chen T., Hsieh JW., Tseng CC. (eds) Advances in Multimedia Modeling. MMM 2011. Lecture Notes in Computer Science, vol 6523. Springer, Berlin, Heidelberg

[10] R. Ajoodha, R. Klein and B. Rosman, "Single-labelled music genre classification using content-based features," 2015 Pattern Recognition Association of South Africa and Robotics and Mechatronics International Conference (PRASA-RobMech), Port Elizabeth, 2015, pp. 66-71. doi: 10.1109/RoboMech.2015.7359500

Enlarged CNN