CS230

Music Genre Classification Using A Convolutional Neural Network

Khalil Miri
Department of Computer Science
Stanford University
kmiri@stanford.edu

Erick Enriquez
Department of Mathematical and Computational Sciences
Stanford University
eenriquez@stanford.edu

Aidan Donohue
Department of Computer Science
Stanford University
aidanjd@stanford.edu

Abstract

In this paper, we will describe our methodology and results for building a music-genre classifier by training multiple deep learning models on the GTZAN dataset (available on Kaggle). We attempt to compare and contrast various models and approaches in order to determine which approach will work best for the task.

1 Introduction

For our final project, we have decided to build a music-genre classifier to predict the genre-composition of a given song. Music genres are helpful for grouping songs and artists to help listeners discover and explore new music. Because of the scope of this project, we decided that, rather than choosing a single model and maximizing our results, we would attempt various approaches to the problem, comparing the results of each approach to decide which model would be the most promising for this task for future researchers. As a result, the inputs to our model are varied; For our first two attempts at the problem, we trained a simple logistic regression model, a standard neural network (NN), and a convolutional neural network (CNN) on grayscale-spectrograms from the GTZAN dataset by vectorizing the images and training them as single-channel images, thus attempting to learn the features of the songs by learning the features of their spectograms, and then outputting a binary output for our logistic regression model and a one-hot vector from our softmax output for both the standard NN and CNN. For our next two attempts, we decided to pass on using the spectogram images and instead extract features from the raw audio using a Fast Fourier Transform and testing on this data instead.

2 Related work

When we began searching the literature for training a music genre classifier, the first question we needed to answer was in what format would we best be able to train on music files. We dove into literature with the preconceived notion that we would be training on raw audio data, but instead found that many papers either made reference to or used fourier transformations to produce spectograms which created images that plotted the pitch of an audio signal against time. This effectively allowed

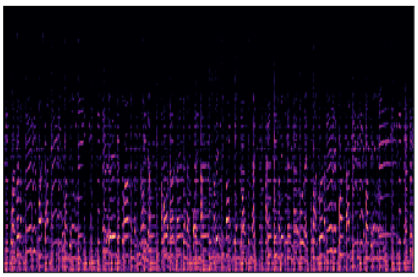
CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

the authors to train CNNs on images which illustrated the audio features rather than on the audio files themselves [9]. While this was true for most of the literature surrounding music-genre classification, another alternative was to use a fast-fourier transform to construct time-series data which could then be trained using Support Vector Machines with various kernels, a machine learning technique that was beyond the scope of this project [1]. One paper, in particular, used a hybrid CNN/RNN model which allowed the model to learn both short-term and long-term temporal data from an audio clip[4]. Though we did not choose to build a hybrid model, the RNN that we used to train on the FFT data we extracted from our audio files was inspired by this model. Because both of these papers, as well as a third which used spectral analysis to learn music features [3] all used the GTZAN Music Genre dataset (which we will describe in more detail in the next section), we decided that this dataset would be suitable for our purposes, as it was lightweight, well-documented, and widely used for this particular task. The final paper that influenced our model and data decisions was one which used spectogram images as well as text-based metadata (which we could not gain access to) to train a standard CNN [2]. This model proved to be state-of-the-art for the task at hand, though we could not replicate it due to constraints on available datasets as well as the team's current skillset.

3 Dataset and Features

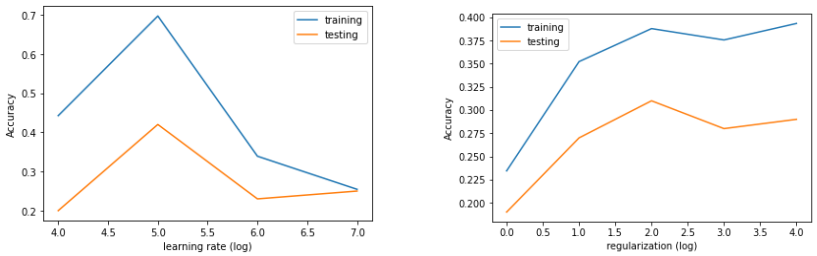
Based on the literature we reviewed, we decided that we would use the GTZAN music-genre dataset available on Kaggle. This dataset contains folders for 10 distinct genres which contains both the raw audio files as well as .png spectograms of the corresponding audio clips for 100 songs from each of the 10 genres. We used both the spectograms and the raw audio to train our models using different approaches for each. For our original logistic regression baseline, we used only a 200 song subset of the the spectograms as a proof of concept for milestone 1, but used the entire dataset to train both our NN and CNN once we had the computational resources to do so. In every case, we trained on 95 percent of the songs, and used 5 percent of them for development. When we built our RNN model, we sliced our audio into three second clips to augment our dataset and then used a fast-fourier transformation to convert them to mel-cepstrum vectors and then used this data to retrain our CNN as well as to train a new RNN-LSTM model.

Figure 1: Example Spectogram from GTZAN Dataset



4 Methods

Although we began our project by training a small logistic regression model on two musical genres with a cross-entropy loss function, we quickly recognized the inherent lack of robustness and ability for this model to adequately learn features for various genres, so we will skip the discussion of this model for the purposes of this paper. Our first real-attempt at training a multi-class model was a standard neural network with a softmax output. For this model, we fed the data from each pixel entry (grayscale) to a 5-layer neural network with a softmax cross-entropy loss-function. In order to decide on this architecture, we experimented with 4 and 7 layer models (while also experimenting with the hidden units in each layer throuhguo) to see how the variance increased as we increased the layers and found that 5 gave us our peak accuracy on the development set, which we used as our metric for each model. This model's variance with all the architectures we tried was quite high so we attempted to implement L2 regularization to see if this would allow us to achieve a higher dev-set accuracy.



(a) Learning Rate (b) Regularization Parameter

Figure 2: Accuracy vs Hyperparameter for Standard NN

Since this did not have much effect on the model, we noted that this model was not robust enough to learn the features of our spectograms. Furthermore, using the same metric, we also attempted to tweak other hyperparameters such as our mini-batch sizes (4, 8, 16, 32) and learning rate (which we tweked by powers of 0.1) and plotted them against our dev-set accuracy in order find the optimal combination of these parameters. Once we established a mini-batch size, we further tweaked the learning rate using powers of 0.5 to find the learning rate that maximized our accuracy.

Once we decided that our simple neural network was insufficient for the task, we tried again to train a model on our grayscale spectograms, but this time with a convolutional 2D network with two RELU layers, two maxpool layers, a flattening layer, and a fully connected layer. The first maxpool layer used an 8x8 filter with an 8x8 stride while the second used a 4x4 filter with a 4x4 stride, both had "SAME" padding. We used the same loss function as our previous model to measure convergence. Using optimized our hyperparameters using the same methods as we did when implementing our standard neural network.

After once again acheiving disappointing results with this model, we decided to change our approach altogether. Rather than training on the grayscales of our spectograms, we instead used a fast-fourier transformation to extract features from our raw audio data and run these features through an RNN-LSTM model to capture the short-term and long-term relationships between features across a piece of audio. In order to extract the the appropriate features for learning, we used the .wav 30 second audio clips and slip them into separate audio clips in 3 second intervals before converting them into a mel-frequency cepstrum (MFCC) using the librosa library which then created a vector of MFCC vectors for each of the thirty audio clips s1, s2, ...s30, thus effectively allowing us to reduce the computational load for each data point. We ran this data through both our RNN-LSTM and CNN models for comparison.

5 Experiments/Results/Discussion

As mentioned earlier in the paper, we had rather lackluster results after experimenting with our first neural network. Even after implementing a grid-search of our hyperparameters to measure our dev-set accuracy across various min-batch sizes, learning rates, and regularization techniques, we found that our model was still overfitting to our dataset. In particular, we noticed that this model was capped at around 90% training set accuracy and 45% development set accuracy. We attempted to implement regularization techniques such as L2 regularization and also played around with techniques such as Adam Optimization and still could not perform better than before. Even after adjusting the architecture of our model once we secured the computational resources to do so, we decided that this model of incapable of surpassing the aforementioned benchmark and began to fear that we may not have enough data to accurately accomplish this task since our data likely only covered a subset of the total features (12000) contained in each example. Reassured by the results achieved in the literature on this exact dataset, we decided to pursue our second model.

It was around this time that we discovered Convolutional Neural Networks both in class and in the literature and decided to train our CNN on grayscale spectograms of our music. Unfortunately, even

after extensive hyperparameter tuning, we plotted our dev-set accuracies (our primary metric) against various hyperparameter values just as before and found that, while this model performed better, we were still suffering from the problem of overfitting our data. This time, our model acheived 99% accuracy on our training set but no more than 64% accuracy on our development set, even with various archiecture modificaitons which resulted in the one described in the previous section. Given this overfitting, we still worried that we may be working with insufficient data, but, given that we could not efficiently gain access to more, we pursued a different route instead.

Rather than training on our grayscale spectograms, we were inspired by the RNNs we learned about in class during this time and decided to pursue a method which leveraged these as well as the powerful and deep CNNs we had already been working on. It was for these reasons, as well as because of further literature review that we decided to build and train an RNN-LSTM model to train on the Fast-Fourier transformations of our raw audio data by converting them into vectors of MFCC vectors. This model, unfortunately, performed very poorly. In order to gauge its performance, we re-trained our CNN on this new dataset for comparison. Aside from taking far longer to train than the CNN on the same data, the model seemed capped at around 55% dev-set accuracy even after tuning, while the CNN achieved a 75% dev-set accuracy on the same data format: our best result so far. We believe this is the result of two primary factors: our dataset size and the temporal relationships between the data. In regards to the former, we believe the increased computational complexity of the model requires far more data to adequately learn the features of our audio clips, even with the aforementioned augmentations. In regards to the latter factor, we believe the temporal significance of the MFCCs don't provide the LST with much of an advatage since the MFCC vectors effectively encoded temporal relationships into the data itself, which the CNN could learn effectively.

6 Conclusion/Future Work

After all was said and done, we attempted to perform music-genre classification using three distinct models and four distinct approaches: a standard neural network with softmax output and a convolutional neural network with softmax output which we trained on the spectograms provided by the dataset, and an RNN-LSTM model which we trained on an augmented, spliced version of our raw audio clips, using this data to retrain the CNN for comparison. The two models we trained on our spectograms performed rather poorly, suffering from high-variance without surpassing a 65% dev-set accuracy threshold, while the third model did not perform much better when trained on our augmented dataset, though this data set did show promise when used to re-train our CNN, which indicates that the MFCC dataset may be a better alternative to the grayscale spectograms for this task. Across all of our models, we had an obvious variance issue with all models achieving a max training-set accuracy between 90 and 100% while remaining in the 60 to 70% range for development-set accuracy. This, combined with teh varying complexity of the models, indicates to us that our project suffered from a lack of data caused by two primary obstacles including a lack of access to data of the same format as well as a lack of computational resources which would have prevented us from reiterating through as many different models and hyperparameters as we could've due to significantly longer run-times. If we were to redo this experiment with more time and computational resources, we would seek to gain access to many many more raw audio clips with genre labels, convert them to MFCCs, and implement an increasingly robust CNN, which showed the most promise given this data format.

7 Contributions

Literature was read and reviewed by all three members of the team. Furthermore, the goals and outline of the project were also discussed by all three members. The details of the implementation and analysis of each of the models was performed independently and then shared with the rest of the team. The paper was primarily authored by Enriquez, but reviewed and edited by Miri and Donohue.

Enriquez: Responsible for the implementation and tuning of the CNN model as well as the primary author of the papers at each milestone. Responsible for distilling and synthesizing findings from all models and trials and consolodating them for the final project paper and video. Primarily responsible for the design and production of final project video.

Miri: Responsible for the implementation of the standard Neural Network as well as error analysis

for both the NN and CNN, as well as for the grid-search hyperparameter tuning performed on both. Reviewer and editor of final paper, secondarily responsible for video design and production.

Donohue: Responsible for the extraction of features from raw audio data as well as the implementation and tuning of the RNN-LSTM model and its comparison with the CNN model on the same data format. Did not obtain images/plots for RNN-LSTM model for video.

References

[1] Elbir, Ahmet, et al. "ResearchGate." ResearchGate, Short Time Fourier Transform Based Music Genre Classification, 2018.

[2] Oramas, S., Barbieri, F., Nieto, O. and Serra, X., 2018. Multimodal Deep Learning for Music Genre Classification. Transactions of the International Society for Music Information Retrieval, 1(1), pp.4–21. DOI: http://doi.org/10.5334/tismir.10

[3] Banitalebi-Dehkordi, M., Banitalebi-Dehkordi, A. Music Genre Classification Using Spectral Analysis and Sparse Representation of the Signals. J Sign Process Syst 74, 273–280 (2014). https://doi.org/10.1007/s11265-013-0797-4

[4] Dwivedi, Priya. "Using CNNs and RNNs for Music Genre Recognition." Towards Data Science, Medium, 13 Dec. 2018, towardsdatascience.com/using-cnns-and-rnns-for-music-genre-recognition-2435fb2ed6af.

[5] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[6] J. D. Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science Engineering, vol. 9, no. 3, pp. 90-95, 2007.

[7] Oliphant, T. E. (2006). A guide to NumPy (Vol. 1). Trelgol Publishing USA.

[8] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E.A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. (2020) SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods, in press.