Deep Music Genre

Miguel Flores Ruiz de Eguino
Stanford University
miguelfr@stanford.edu

Abstract

In this report I present an approach for automatic music genre detection and tagging using convolutional neural networks. I evaluate different architectures using the GTZAN and MagnaTagATune datasets using the song clips melspectograms as input to the convolutional neural network. I present the best results and the architecture with which I managed to obtain those results. Future work could involve visualizing each layer of the neural network and also experimenting with more architectures to improve the accuracy.

1. Introduction

Music genre classification is a popular problem in machine learning with many practical applications. One application could be in music recommendation. The neural network learns the features of a song that makes it more likely or less likely to belong to one genre or another. Then it's able to classify its genre (or sub-genres) automatically and hence once we understand the song genre, we can use that information for music recommendation and discovery. Another application could be to automatically organize a huge music corpus and tag every song by genre, sub-genre or other tags like the instrument being played, whether or not there are vocals in the song, etc. These data can be used for finding similar songs.

Convolutional neural networks are known to give great results in the area of computer vision. For some years, we have seen a lot of progress in this area until even reaching better than human accuracy for image classification. These neural networks consist of a convolutional layer followed by pooling layer. These networks learn to recognize different features of the input and when stacked one after each other, more complex features are learned. Some optimizations have been introduced through the years, like dropout to avoid overfitting, batch normalization to make weight initialization not an issue, etc. In this project I'll be using mainly convolutional layers, max pooling layers, average pooling layers. I also used batch normalization and dropout, but they didn't seem to help much in this case. In this case, we'll use neural network ideas for computer vision but applied to music classification, on a spectogram instead of an image.

In this project I;ll use two popular datasets for this task. The first of them, GTZAN, maps songs to 10 different genres. This dataset is small, but used in many papers, so I decided to give it a try. The other dataset is MagnaTagATune. This dataset contains more songs and many labels. These labels are not only genres but also features of the song like whether it has drums, guitar, voice, is a happy song, etc. As explained later in section 3, only 50 tags will be used in this report.

2. Related work

There's some existing work on using neural networks (not only convolutional, but also simply fully connected and recurrent netowkrs) for music genre classification, among other non-deep learning approaches that I won't talk about this time. I'll focus on different methods I read about that use mainly fully convolutional neural networks and also recurrent neural networks. ost of the work I read about use the datasets: GTZAN [21] dataset, the Million Song Dataset [3] and the MagnaTagATune [15] dataset.

Aaron et al. [1] use MFCC spectograms to preprocess the songs. This work doesn't focus on genre recognition, but on song similary for music recommendation. However, I thought it was worth mentioning for their use of convolutional neural networks with ReLU activation on song clips preprocessed as MFCC spectograms.

Tao [7] shows the use of restricted boltzman machines and arrives to better results than a generic multilayer neural network by generating more data out of the initial dataset, GTZAN. In this paper a data distribution problem in the dataset is explained and it shows that it makes it hard to accurately classify more than 4 classes using only the GTZAN dataset. For song preprocessing, this paper suggests the use of MFCC spectograms as well.

Gwardys et al. [8] show an interesting approach involving transfer learning. They initially train the model on ILSVRC-2012 [18] for image recognition and then reuse

the model for genre recognition on MFCC spectograms. The architecutre used in this article consists of five convolutional layers, the first two and the last one with max pooling as well. In the end, three fully connected layers.

A popular article on recommending music at Spotify [5] shows the use of convolutional neural networks for music genre classification. This article uses an architecture that consists of 3 convolutional plus max pooling layers and finally max pooling, average pooling and L2 pooling concatenated and fed into three fully connected layers. The article uses melspectograms for song preprocessing, which seems to be the standard approach for song clips and as of now, giving the best results.

There are other similar approaches that use fully convolutional neural networks for this problem. These approaches use fully convolutional neural networks. These architectures consist of a convolutional layer followed by a max pooling layer N times and finally a fully connected layer [13][11][17][19][20][12]. All those articles have minor differences on the number of layers, hyperparameters, etc. But in the end, the idea behind them is to use fully connected neural networks.

Keunwoo et al. [14] present an approach using convolutional recurrent neural networks. In this approach, the output of a convolutional neural network is fed into a recurrent neural network and finally into a fully connected layer.

From all these works, the representation of the song that seems to work the best is melspectograms. As mentioned above, MFCC spectograms has good performance too, but usually melspectogram representation beats it.

3. Dataset and features

Given that music is copyrighted, coming up with a good dataset is quite complex. The three datasets that seem to be the most popular for music genre classification are: GTZAN, MagnaTagATune and the Million Song Dataset, as I mentioned before.

Originally I was planning to use the Million Song Dataset and start with its subset of 10k songs. However that dataset doesn't include audio, only song metadata. I've written a script to fetch the audio of it since one of the metadata fields it contains is a 7digital song id. Out of 3242 samples, only 622 were available in 7digital. So some dataset balancing would need to be done first if I want to use that subset. Another limitation is that their API only allows 4000 requests per day, so downloading the whole Million Song Dataset would take many days without setting up many accounts.

I started using the GTZAN [21] dataset. This dataset consists of only 1000 songs and 10 genres. I found that the small size of the dataset makes it hard to converge when using deeper models. For this dataset I generated the melspecotgram for all the songs and serialized all the melspectograms as a numpy[10] array. Later this data was loaded in memory for training. Given the small size of the dataset, it was simple to load the data into numpy arrays in memory and directly fed into the Keras model fit method. For the labels I used a one-hot vector where 1 is the expected genre of the song.

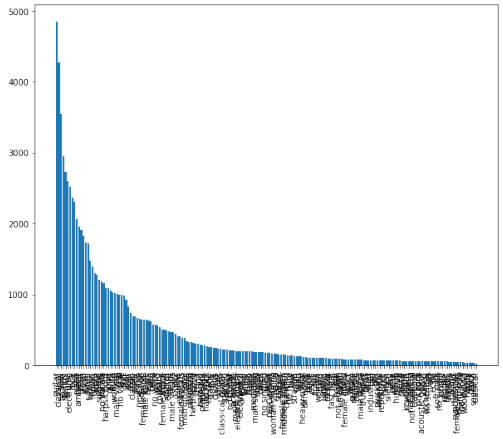

Figure 1. MagnaTagATune tag distribution

The dataset I ended up using was the MagnaTagATune dataset. This dataset consists of 25863 song clips of 29 seconds each and 188 tags for each song. These tags are the instruments in the song, the genre, whether or not it has vocals, the mood, among other tags. Not only genres. Sadly this dataset is not balanced as Figure 1 shows where the tags are in the X-axis and the number of songs that have that tag in the y-axis.

I followed the approach that many of the papers I came across were using, which consists on picking the top 50 tags and use only the songs that include those tags [13]. By doing this I endeded up with a training dataset of 13510 songs, a test dataset of 4223 songs and a validation dataset of 3378 songs. In this case I couldn't load all the data in memory (something I tried initially, but I was running out of memory), so I ended up using Tensorflow 1.2rc0 [2] data API to load the dataset. Later I changed the approach of saving and loading the songs to use TFRecords. This Tensorflow format represents a sequence of binary strings. According to the docs, the format is useful for streaming large amounts of data sequentially. Therefore, each song was saved as a TFRecord file that had the song melspectogram and its label. The labels are represented as vectors of zeros and ones where one means that the song has the tag associated to that index.

Initially all the songs are prepocessed as melspectograms (see Figure 4). Computing the spectograms makes extensive use of the librosa [16] library for audio processing. The window size was set to 2048 and the mel and the frequency bins to 128. The spectograms are then normalized by sub-

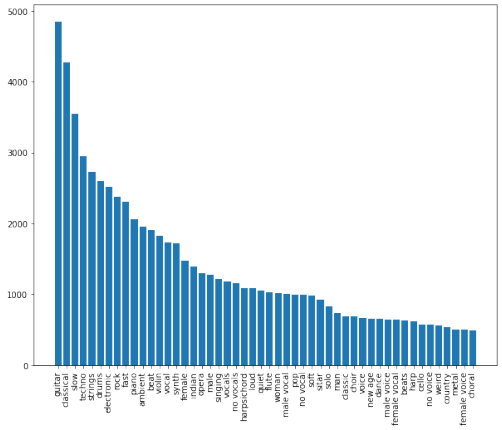

Figure 2. MagnaTagATune tag distribution for the top 50 tags

```
import librosa
import numpy as np
y, sr = librosa.load(song_path, mono=True)
spectogram = librosa.feature.melspectrogram(
    y=y,
    sr=sr,
    n_mels=128,
    n_fft=2048,
    hop_length=1024)
spectogram = librosa.power_to_db(
    spectogram, ref=np.max)
```

Figure 3. Computing melspectogram with librosa

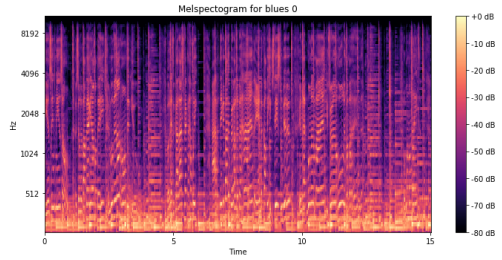

Figure 4. Melspectogram for a blues clip

tracting the mean and dividing by the standard deviation. Using librosa, the melspectogram is computed as shown in Figure 3.

4. Methods

For GTZAN, the architecture used is based on the one proposed in [5] with small modifications. Initially the network has three convolutional layers with 256 filters each of size 4 and stride 2. Each layer has a ReLU (see Equation 1) activation after which a max pooling layer wiht pool size 2 comes. The convolution and pooling is done in 1D in

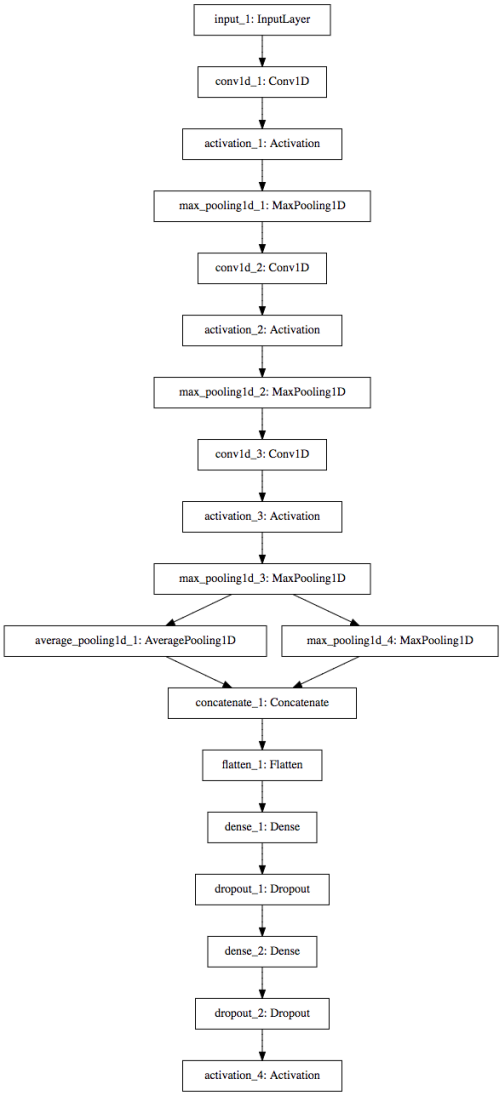

Figure 5. Model used for GTZAN

the time dimension, not in the frequency. After these layers a max pooling and average pooling layer come in parallel, each of size 4. The output of these layers is then concatenated and fed into a fully connected layer of size 2048 and finally the 10 classes.This 2048 layer has dropout and ReLU activation. The last layer has softmax activation and cross entropy loss was used.

See Figure 5 for the network that was implemented on Keras [4].

For this model I also experimented using a Resnet [9] architecture, but I didn't get that great results, they were usually around 20% accuracy only.

$$ReLu(x) = \max\{0, x\} \tag{1}$$

For MagnaTagATune, this model didn't work that good.

I tried some approaches using variations on the hyperparameters, more layers, batch normalization, dropout, 2D convolution instead of 1D, etc. But I was not able to reach more than 22% accuracy using that model and those variations. The best results I was able to get were with a fully convolutional neural network[13].

Unlike GTZAN, I obtained better results when using 2D convolutions instead of 1D. The architecture that gave me the best results consists of a convolutional layer followed by a max pooling layer three times. Each convolutional layer has a kernel of size 3, ReLU activation and 128, 256, 512 filters respectivley. The max pooling layer has pool sizes of 3x4, 4x5 and 5x6 respectively (these non-squared shapes for the pooling, were inspired by [13]). Finally a fully connected layer. Dropout didn't help much and neither did batchnormalization between the convolutional layer and the maxpooling layer. Sigmod crossentropy loss was used (See Equation 2 for the Tensorflow implementation [6] where x are the logits and z the labels).

See Figure 6 for the network that was implemented on Keras [4]. I show the network from Keras here, but I ended up implementing this model directly on Tensorflow since it was easier to load that bigger dataset using the new dataset API from Tensorflow 1.2rc0.

$$\max\{x, 0\} - xz + \log\left(1 + e^{-|x|}\right) \tag{2}$$

5. Results

For GTZAN, the dataset is splitted into 33% for validation and the rest for training. Training is done using Adam optimizer with learning rate of 0.001 and decay of 0.01. After training like this, the model reaches an accuracy of 56.12% in the training set and 58.79% in the validation set. Figure 7 shows how the classes are currently classified. Figure 8 and Figure 9 show the loss and accuracy during training of this model.

For MagnaTagATune, the dataset is splitted into 20% for test and 80% for training. 20% of of the training dataset is used for validation. Training is done using Adam optimizer with an initial learning rate of 0.0001 and exponential learning rate decay. After training like this for 20 epochs, the model reaches and accuracy of 48.48% on the training set (See Figure 10) and of 46.96% on the validation set.

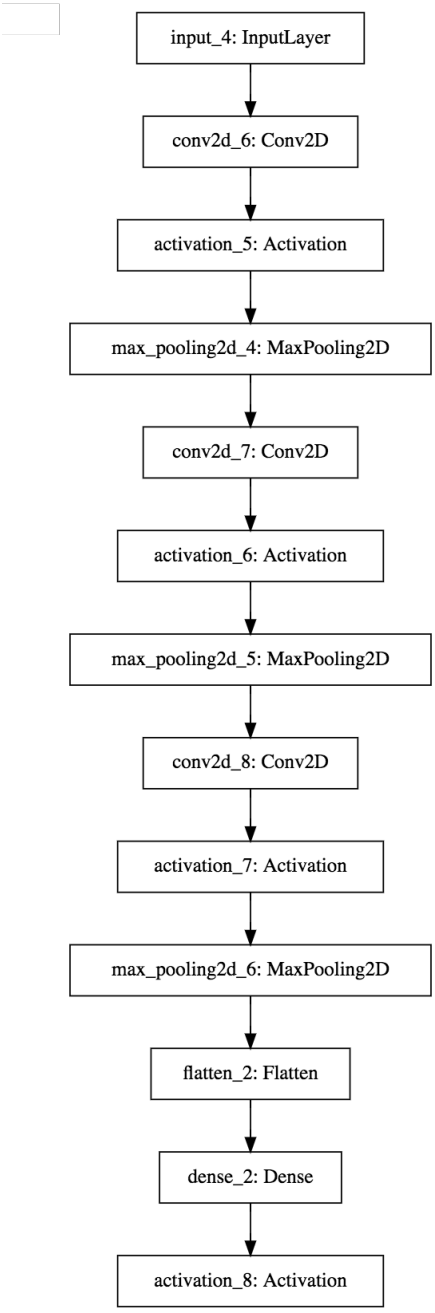

Figure 6. Model used for MagnaTagATune

6. Conclusion and future work

I managed to get decent accuracy on the datasets I worked with. However, that accuracy is not the state of the art for the datasets used. Even though I used very similar architectures, I couldn't reproduce the reported accuracies of more than 85% in MagnaTagATune as well as in GTZAN. Probably I didn't train for enough epochs, but training was quite slow particularly on MagnaTagATune. However, this was a good way to practice exploring different architectures, setting up the number of layers, trying different hyperparameters and debugging the neural network training.

Something I wanted to try but in the end I didn't have time is to use saliency maps to visualize the different filters the neural network is learning for the songs.

Another thing I wanted to explore but didn't have time either was the use of LSTMs to see how the model per-

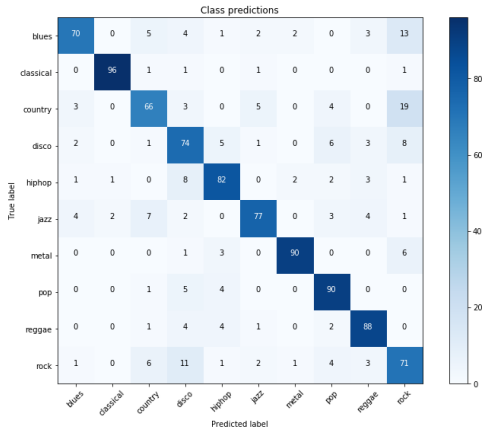

Figure 7. Confusion matrix of genre predictions

Figure 8. Accuracy during training GTZAN

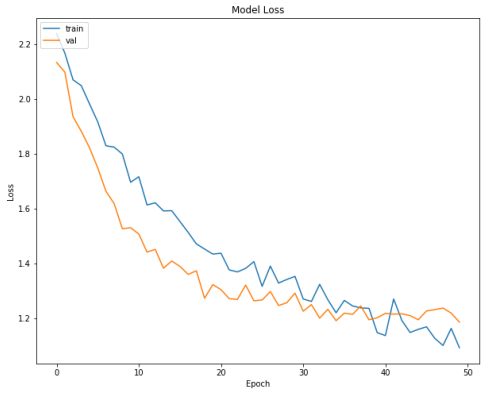

Figure 9. Loss during training GTZAN

ception of the song genre changed over time as the song progressed.

Another thing that I'd like to explore at some point is neural style applied to music instead of pictures. Try to give a rock n roll feel to a hip hop song or viceversa for example.

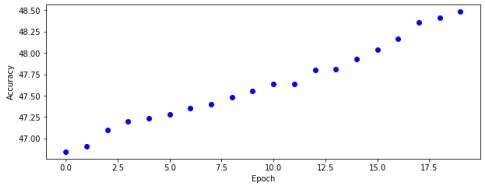

Figure 10. Accuracy during training Magnatagatune

Music generation is another topic that I find quite interesting and that I'd like to explore at some point. Some ideas from here could probably be used.

References

[1] B. S. Aaron van den Oord, Sander Dieleman. Deep content-based music recommendation. 2013.

[2] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. J. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Józefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. G. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. A. Tucker, V. Vanhoucke, V. Vasudevan, F. B. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *CoRR*, abs/1603.04467, 2016.

[3] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

[4] F. Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.

[5] S. Dieleman. Recommending music on Spotify. `http://benanne.github.io/2014/08/05/spotify-cnns.html`, 2014. [Online; accessed 15-May-2017].

[6] T. docs. SciPy: Open source scientific tools for Python, 2017–. [Online; accessed 06/01/2017].

[7] T. Feng. Deep learning for music genre classification. 2014.

[8] D. G. Grzegorz Gwardys. Deep image features in music information retrieval. 2014.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[10] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [Online; accessed 06/01/2017].

[11] J. N. Jongpil Lee. Multi-level and multi-scale feature aggregation using pre-trained convolutional neural networks for music auto-tagging. 2017.

[12] K. L. K. J. N. Jongpil Lee, Jiyoung Park. Sample-level deep convolutional neural networks for music auto-taggin using raw waveforms. 2017.

[13] M. S. Keunwoo Choi, Gyorgy Fazekas. Automatic tagging using deep convolutional neural networks. 2016.

[14] M. S. Keunwoo Choi, Gyorgy Fazekas. Convolutional recurrent neural networks for music classification. 2016.

[15] Magnatagatune. Magnatagatune. `http://mirg.city.ac.uk/codeapps/the-magnatagatune-dataset`. [Online; accessed 7-Jun-2017].

[16] B. McFee, M. McVicar, O. Nieto, S. Balke, C. Thome, D. Liang, E. Battenberg, J. Moore, R. Bittner, R. Yamamoto, D. Ellis, F.-R. Stoter, D. Repetto, S. Waloschek, C. Carr, S. Kranzler, K. Choi, P. Viktorin, J. F. Santos, A. Holovaty, W. Pimenta, and H. Lee. librosa 0.5.0, Feb. 2017.

[17] Y. B. D. E. Philippe Hamel, Simon Lemieux. Temporal pooling and multiscale learning for automatic annotation and ranking of music audio. 2011.

[18] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[19] D. P. W. E. J. F. G. A. J. R. C. M. M. P. D. P. R. A. S. B. S. M. S. R. J. W. K. W. Shawn Hershey, Sourish Chaudhuri. Cnn architectures for large scale audio classification. 2017.

[20] A. B. C. Tom LH. Li and A. H. Chun. Automatic musical pattern feature extraction using convolutional neural network. 2010.

[21] G. Tzanetakis and G. Essl. Automatic musical genre classification of audio signals. In *IEEE Transactions on Speech and Audio Processing*, pages 293–302, 2001.