Music visualization with end-to-end learning

Bruno Godefroy

May 3, 2017

1 Abstract

Music visualization is nowadays very commonly used in media player softwares, such as *Winamp* or *Itunes*. These systems rely on the extraction of features from the raw audio, such as spectrograms, pitch or timbre. Then, preprogrammed models, called "presets", define how animations behave, depending on these features [1].
The features commonly used are usually hand-crafted and depend on some historical background. In recent years, there has been increasing interest in using features learning and deep architectures instead, thus reducing the required engineering effort and the need for prior knowledge. It has been shown that these new techniques could create some more effective representations and improve algorithms performance in music classification and music auto-tagging [2].
Since autoencoders provide the best possible representation of some input data, this could obviously present a great interest for visualization as well. In this project, we propose to experiment some music visualization based on features extracted with a stacked autoencoder neural network.

2 Method

For this project, we will need first to train the autoencoder model, and then use its output as input parameters for some 3D animation.
The autoencoder neural network could be fed with various input data, extracted from the music frame. It has been shown in previous researches that features learning is more efficient on mid-level representations, such as spectrograms, than on raw audio data [3]. Thus, the input and output of the network will be the audio spectrogram of the current frame.

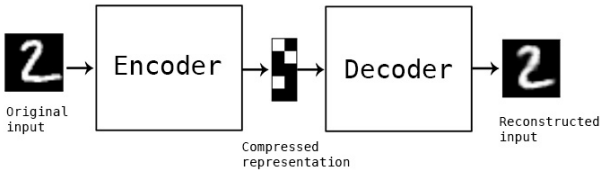

Figure 1: The autoencoder framework

Then, depending on the number of parameters we need for visualization, the size of the hidden layer is defined. For example, if we want to create a particle system visualization, about 20 parameters need to be tuned for each music frame (position, color, wind, gravity...). In this case, we would use a hidden layer of size 20, so that the autoencoder generates a representation of the input spectrogram using 20 features.
The last remaining task consists in mapping the features to the visualization parameters. This could be done randomly, given some lower and upper bounds for each parameter. If this doesn't produce satisfying results, this matching could be done manually.

3 Implementation

The aim of the project is to create a real-time visualization Web app, such as [4]. Once launched, the user could either provide an audio file or load a default sample and then visualize a generated animation

while the music is playing. The particle system is probably the easiest visualization to create and could provide a very good feedback of the method. Therefore, it will be done first and maybe some other animations will be added later.
Since we want to mainly focus on the visualization part, we will try to handle the machine learning task quickly, using some libraries, without going into further details. In practice, we will use the popular library *Caffe*, first in local for the training part (to maximize performance), and then its JavaScript extension *Caffe JS* to run the model into the client Web browser.
For the visualization, we use *Three.js* with *WebGL* and maybe a few libraries, such as the particle system provided in the assignments (animation track).

References

[1] Robyn Taylor, Pierre Boulanger, and Daniel Torres. *Real-time Music Visualizations using Responsive Imagery.*

[2] S. Sigtia and S. Dixon. *Improved music feature learning with deep neural networks.* In Proceedings of the 38th International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2014.

[3] Sander Dieleman and Benjamin Schrauwen. *End-to-end learning for music audio.* Conference Paper in Proceedings - ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing, May 2014

[4] Felix Turner. *Loop Waveform Visualizer.* https://airtightinteractive.com/demos/js/reactive