Conditional SampleRNN - Generating Emotional Music

Vincent Brisse
London, United Kingdom
vab18@ic.ac.uk

ABSTRACT

SampleRNN is a deep-learning audio generation model. The model is autoregressive: it generates one sample at a time. Its modular architecture, combining memory-less modules and stateful recurrent neural networks in a hierarchical structure, is the key to learn the important structure at many time-scales of raw audio signal.

This thesis presents the design and implementation of a conditional SampleRNN, which is then used to generate music conditioned on emotion. The model is implemented with PyTorch. Experiments show that the conditional model works but that it struggles to capture emotion. Both the code[1] and generated samples[2] are available online.

INTRODUCTION

Art and emotions are deeply interlinked and since recent deep learning models have been able to generate artworks and musical pieces it is interested to experiment and find out if current models are able to capture emotion and to turn emotion into art. *The Emotional GAN* [1] - which inspired this paper - proposes a way to turn emotion

To generate audio using the raw waveform is a challenging task. The minimal sample rate for audio files is 16 kHz which means the model has to output around 3,000,000 samples for a 3-minute long music track. Moreover, the model has to figure several levels of abstraction: short timescale (timbre of a voice), medium timescale (phonemes) and long timescale (sentences).

SampleRNN is an *unconditional end-to-end neural audio generation model* [4]. Its architecture is based on a hierarchy of modules operating at different resolutions of the audio-signal. The upper modules work with large chunks of samples and are based on Recurrent Neural Networks (RNNs), while the lower module is a multilayer Perceptron (MLP) which works with a few samples and the conditioning from the module above it and outputs the probability of the next sample.

[1] https://github.com/VincentSample/Conditional-SampleRNN/
[2] https://soundcloud.com/sample-rnn/sets

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

The original SampleRNN model is *unconditional* which means it is not able to learn extra useful information (such as emotion) about the data it is trained with.

In this paper we propose an architecture for a Conditional SampleRNN and its PyTorch implementation and we experiment with different datasets to find out if the model is able to capture *emotion* in music.

RELATED WORK

While it has been made clear quite a while ago that humans could not match AI on purely logical activities (Deeper blue vs Kasparov (1997) or AlphaGo (2015)), the jury is still out on whether or not AI can match human creativity. Some people argue that AI and especially AI based on Deep Learning techniques cannot be *creative* because the model is trained using previous artworks. However, while not creative *per se*, machine learning can at least be used as a tool in the creative process.

Although no AI-generated music piece has hit the charts just yet, it is undeniable that impressive progress has been made recently in music generation. In the symbolic domain representation, Music Transformer has been able to produce some coherent melodies[3].

In the scope of this thesis is *raw audio synthesis*: generating directly each point of the raw waveform (at least 16,000 samples a second). Autoregressive models have proven to be successful with raw signal. An *autoregressive* model is a sequential model where the prediction of the sample at timestep T depends on all previous samples at timesteps $[1,...,T-1]$:

$$p(X) = \prod_{i=0}^{T-1} p(x_{i+1}|x_1,...,x_i) \tag{1}$$

Autoregressive models were thought to be infeasible due to the complexity of the problem but in 2016 a Google DeepMind team proposed WaveNet[4] [7], a neural network based on dilated causal convolution which was able to generate good quality samples, but the model is both expensive to train and to generate from. Since then WaveRNN [3], an architecture based on Recurrent Neural Networks, produced similar or better results but with a much faster generating time. But the model is difficult to understand, difficult to reproduce and training time is still very long. SampleRNN [4] is a solution

[3] https://magenta.tensorflow.org/music-transformer
[4] https://deepmind.com/blog/wavenet-generative-model-raw-audio/

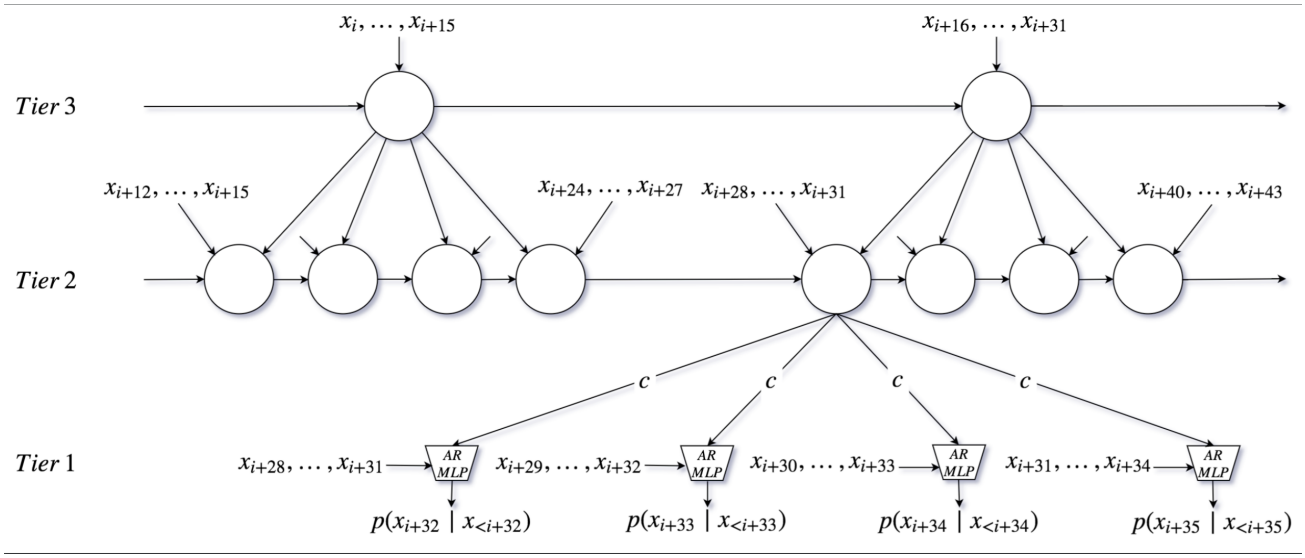

Figure 1. The architecture of the reference SampleRNN module with three tiers, frame sizes $FS^{(3)} = 16, FS^{(2)} = 4, FS^{(1)} = 4$. Upsampling ratio $r = 4$ in each tier. Only one RNN per tier is shown for simplicity's sake. Source

.

to all those problems : the architecture proposed is intelligible, training time is slow but manageable using cloud computing, and generating time is faster. Since we want to capture emotion in music we first have to create a *conditional* architecture.

There are two kinds of conditioning and the WaveNet architecture allows both kinds [7]. *Global* conditioning enables the generation to be conditioned on an information that is fixed across all timesteps such as the music genre. *Local* conditioning is when the model is conditioned with another timeseries such as notes or instruments being played at different timesteps.

MODEL

SampleRNN

The SampleRNN's architecture is based on a hierarchy of modules working at different timescales (see figure 1). There is an arbitrary number K of tiers in the model and each tier takes as input a "previous samples" vector of a different size. The bottom tier called the *Sample-level Module* is an MLP that outputs a probability distribution over the next sample. Each tier above it is a *Frame-level Module*, an RNN that works with large chunks of data and that outputs a *conditioning vector* which becomes the input of the tier below it. In this paper, Gated Recurrent Units (GRUs) have been preferred to Long Short-Term Memories (LSTM) units as the memory cells in the RNN.

Frame-Level module

At a fixed timestep T, a *Frame-level Module* at tier k with a frame size $FS^{(k)}$ takes as an input the previous $FS^{(k)}$ samples and the upper tier conditioning vector $c_t^{(k+1)}$ which is the output of the tier above it $(k+1)$. For the topmost tier, the input simply is a vector of previous samples of size $FS^{(K)}$. Since each module operates at a different resolution with a different frame size, the output conditioning vector needs to be *upsampled* before it is fed into the next module downward. In this paper, the upsampling has been done with transposed convolution layer instead of a linear projection in the reference paper.

$$input = \begin{cases} W_x f_t^{(k)} + c_t^{(k+1)}; & \text{if } 2 \leq k < K. \\ f_t^{(K)}; & \text{if } k = K. \end{cases} \quad (2)$$

$$h_t = GRU(h_{t-1}, input) \quad (3)$$

$$c_{t*1}^{(k)}, ..., c_{t*r}^{(k)} = ConvTranspos1D(h_t) \quad (4)$$

Where k is the tier, t is the timestep, h_t is the hidden state memory of the RNN, $f_t^{(k)}$ is the vector of input samples of length $FS^{(k)}$ and r is the upsampling ratio $(FS^{(k)}/FS^{(k-1)})$

Sample-level Module

The *Sample-level Module* is the bottom tier module. Its input is a concatenated vector of the $FS^{(1)}$ previous inputs with the upper tier conditioning vector $c_t^{(2)}$. It outputs a probability distribution over the next samples. Since this module's frame size is small, a MLP has been preferred to an RNN to speed up training.

$$input = W_x f_t^{(1)} + c_t^{(2)} \quad (5)$$

$$p(x_{t+1}|x_1, .., x_t) = Softmax(MLP(input)) \quad (6)$$

Truncated Back Propagation Through Time

As with standard RNNs, the model is trained with back propagation through time (BPTT). But with high-dimensional data such as audio, BPTT is very computationally expensive and requires a huge memory bandwidth. To overcome this problem the model is trained with *truncated back propagation through time*. The audio files are first split into *subsequences*

of a few milliseconds before being used as training data in the network. Experiments show that the model is still able to learn long-term dependencies (thanks to the conditioning vectors). For more information about the model, TBPTT and the way the samples are formatted, the reader is invited to read the reference SampleRNN paper [4].

Conditional Model

To generate emotional music the model needs to learn what an emotion is. A naive way to do so would be to train the model n times (with n being the number of emotions the model needs to learn) for each emotion $C \in n$ with the dataset related to this emotion. But this is not ideal, for example if we condition voices on speaker identity, the model would have to learn each time what a *voice* is before it can learn what the *speaker identity* is. Also if we conditioned on two classes (*e.g.* both *speaker identity* and *language*) the model would learn both at the same time and would not been able to dissociate the two characteristics.

In this paper, global conditioning is set up by concatenating a vector representing the class to the samples at training time. The class is encoded in a one-hot vector. For example if we have 4 classes, the first class is represented with $1,0,0,0$, the second class is represented with $0,1,0,0$ and so on. The concatenated vector of previous samples and one-hot encoded class vector is then added with the upper tier conditioning vector and fed as the input to all the *frame-level modules*. The *sample-level module* is not modified from the reference model: the class is already encoded in the upper tier conditioning (see figure 2).

$$input = \begin{cases} W_x(f_t^{(k)} ++ C) + c_t^{(k+1)}; & \text{if } 2 \leq k < K. \\ (f_t^{(K)} ++ C); & \text{if } k = K. \end{cases} \tag{7}$$

$$h_t = GRU(h_{t-1}, input) \tag{8}$$

$$c_{t*1}^{(k)}, ..., c_{t*r}^{(k)} = ConvTranspos1D(h_t) \tag{9}$$

Here $++$ is the concatenate operator, and C is the one-hot vector representing the class.

Generation

When generating, the model is primed with a sequence of $0s$ and the class C we want to generate samples from. Samples are generated one by one as the forward pass is run repeatedly at each time-step. The generated sample is picked using the coefficients of the output of the bottom tier: a multinomial distribution over x_{t+1}.

DATASETS

Intuitively, gathering a dataset of raw audio music to induce emotions should not be an issue. Browsing YouTube for "Uplifting music" or "Sad music" leads to hours and hours of results and this is one of the reason why working with raw audio is interesting.

The paper *You Can't Play a Sad Song on the Banjo* [2] shows that instruments are directly linked to emotions: the cello is linked to sad feelings while the banjo is linked to positive emotions, and so even if the same melody is being played. It is the same way with music genre: metal music will always be linked with high arousal emotions (tense, excited), while blues music is linked with low arousal emotions (tired, calm). Moreover, in most songs, different emotions will be felt at different time steps.

However we want the model to capture *emotion* and not instrument nor genre and therefore the model needs to be trained with a specific dataset.

First the conditional SampleRNN model has been tested with a voice dataset conditioned on speaker identity[5] and a mixed piano and violin dataset conditioned on instrument[6]. Then it is trained with various music datasets.

The *Evaluation of musical features for emotion classification*[7] [6] is a dataset gathered from last.fm thanks to the tags labelling the tracks. The most popular musical pieces associated with the following tags have been regrouped: *angry, happy, relax* and *sad*.

The *Emotify* dataset [5] contains 400 1-minute royalty-free songs from 4 different genres, *pop, rock, classical* and *electronic*. It was collected through a game where users were asked to judge their level of *amazement, solemnity, tenderness, nostalgia, calmness, power, joyful activation, tension* and *sadness* throughout the songs. We averaged the user ratings over each songs and then encoded the genre in a one-hot vector and the emotions in a scale vector where 3 means that the emotion is felt strongly and 0 means that the emotion is not felt at all. Both vectors are then concatenated in a 13-long vector - 4 classes and 9 emotions - and fed into the network alongside the samples. Since the condition is a bit different in this case the architecture had to be modified accordingly.

Finally the model was trained on ambient music. Ambient is a *genre of music that emphasizes tone and atmosphere over traditional musical structure or rhythm*[8]. Since ambient music does not use specific instruments and focuses on the atmosphere - which is closely related to the emotions induced by the music - the genre is well-suited for the task at hand. We have chosen to gather a dataset with six hours of audio in four different emotions: *happy (uplifting), sad (melancholy), space (emptiness), and creepy (scary)*[9].

With every dataset, the classes were balanced before training (same amount of audio in each class) and 80% of the data was used for training, 10% for validation and 10% for testing.

[5] https://soundcloud.com/sample-rnn/sets/voices
[6] https://soundcloud.com/sample-rnn/sets/violin-piano-training
[7] https://code.soundsoftware.ac.uk/projects/emotion-recognition/repository
[8] https://en.wikipedia.org/wiki/Ambient_music
[9] https://www.youtube.com/channel/UCoX7b62AL4g7UbQeXOAhZ9Q/playlists

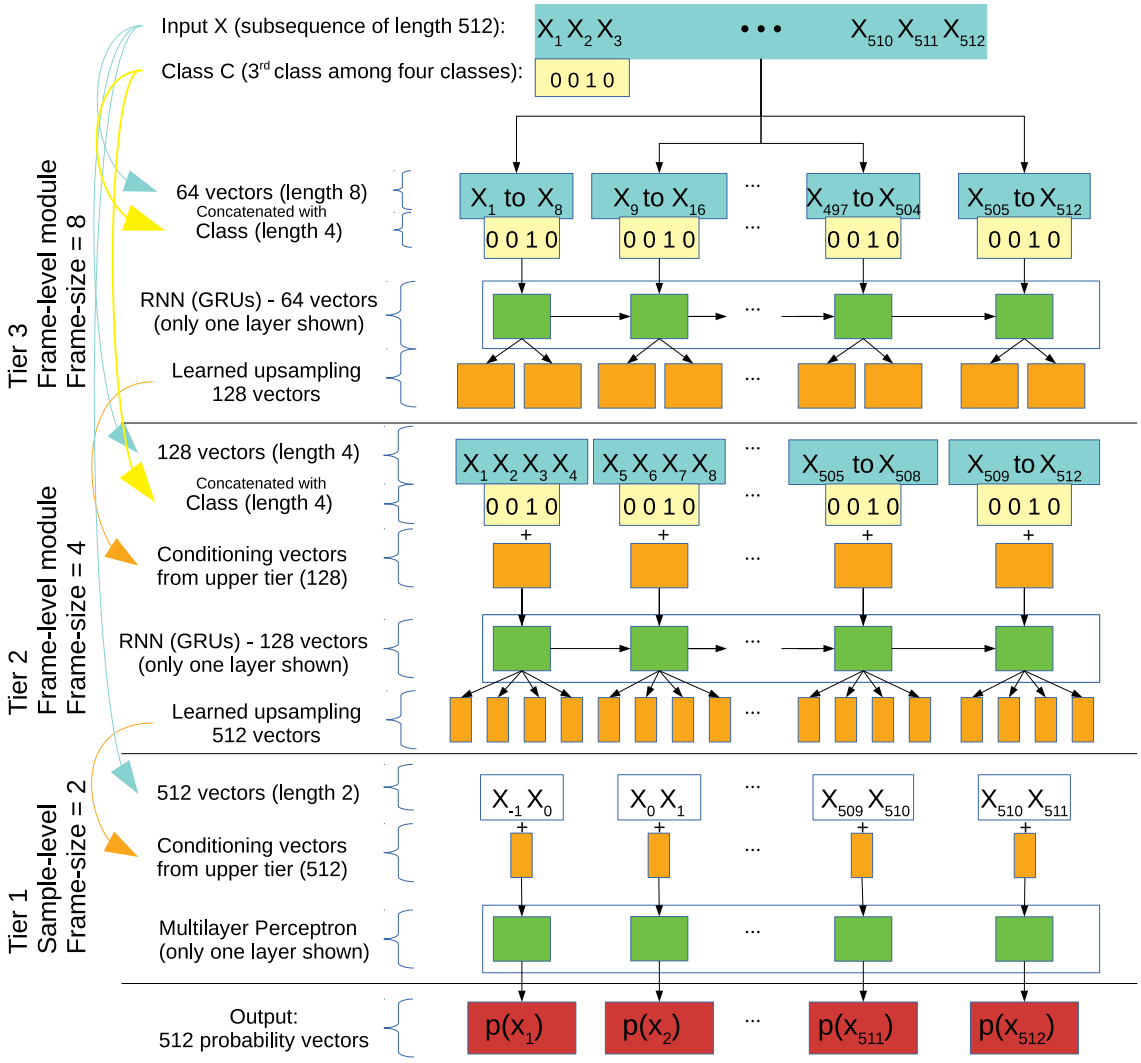

Figure 2. The architecture of the conditional SampleRNN module with three tiers, a subsequence length of 512, and frame sizes $FS^{(3)} = 8, FS^{(2)} = 4, FS^{(1)} = 2$. The model is training on an audio file belonging to the third class of a datasets that contains 4 different classes. In reality, bigger frame sizes are chosen.

EXPERIMENTS AND RESULTS

Experimental setup

With each dataset, the model was trained around 24-30 hours with an NVIDIA Tesla P100 GPU using Google Cloud Virtual Machines. The audio files were first preprocessed into 8-second long .wav files with a sample rate of 16 kHz mono 16 bits.

Since we had no reason to believe that the new conditional architecture's optimal parameters would be different to the unconditional model's optimal parameters and that the unconditional model had already been tested extensively in the original pape, the same optimal hyperparameters were used. A three-tier architecture model was used - it supposedly better capture long-term structure than a two-tier architecture. The top tier frame size was 64, and both the middle tier frame size and the bottom tier frame size were 16. Gated Recurrent Units were used with 1 hidden layer in each RNN and 1024 dimensions in each layer. The batch size was 128 and the subsequence was 1024 samples long. The models were trained with stochastic gradient descent to minimise the Negative Log-Likelihood (NLL) and the update rules from the default Adam optimizer were used.

Results and evaluation

Evaluating a generative model is notoriously hard. The NLL loss function is not strongly correlated to the quality of the audio produced. To evaluate music or art generation researchers tend to prefere human evaluation: it is both expensive and time consuming and has not been done here.

Many samples generated are only noise. In the same generation batch, some samples can be great while others are indistinguishable from random noise. There are also many samples that look good at the beginning and then suddenly crash and become random noise. For this reason, generation was limited to 80,000 samples or 5 seconds.

The voice dataset was used to test the conditional SampleRNN model and it showed that it worked great. The model only trained for one epoch. Of course, since there is no local conditioning on phonemes, the voices sound at best like mumbling but they are discernable from one another[10]. A comparison of audio signals is presented in figure 3. The piano-violin mixed dataset also showed that the model was able to recognize different instruments though it did not produced state-of-the-art results.

The model trained with the dataset gathered from last.fm did not produce great results. The reason is that the songs were from vastly different genres, using a plethora of instruments. However the samples produced were not random noise and can be interpreted as a "mean" of every song it trained on.

The emotify-trained model generated only noise. The reason is that there are too many emotion categories and too many genres and not enough samples in each genre/emotion.

The model trained on ambient music undoubtedly produced the best results[11]. Emotions are recognizable and some samples are of great quality. However many of those samples were noisy and some classes trained much better than others. When we tried to generate longer samples with this model we discovered that the generation did not produce stable samples. Most of them crashed into noise before 30 seconds.

Custom conditioning

We experimented with custom conditioning to see if the model was able to combine emotions. With the best ambient music model, we generated with a conditioning vector $[0,0,1,1]$ which should combine both classes that produced the best samples: creepy and space. This experiment produced only noise, showing that the model is not able to combine classes.

DISCUSSION

Working with raw audio is difficult because of the high-dimensional data. No more than three years ago it was thought unfeasible. Today autoregressive models produce good samples considering the difficulty of the task. Of course models using score representation produce much better melodies than raw audio models since those models only have to focus on the dependencies of the notes and note on the timbre of the audio itself.

But models using directly the raw signal have the great advantage of working with any wave forms it's trained with. In this paper we used this to our advantage by training the model with voices, instruments and ambient music.

To capture emotion, the ideal dataset is monophonic, with an instrument able to generate different emotions (the piano comes to mind), long enough, consistent in the sound quality and the style (classical for example) and annotated regularly with arousal and valence. This dataset does not exist at the moment. The lack of a good and consistent annotated dataset was in the end the main issue in the experimental phase.

Other experiments with the model

A SampleRNN architecture able to be trained with *local conditioning* would be interesting and ideal, but the lack of annotated dataset deterred us to do it.

New experiments could be tried out to test Conditional SampleRNN. For example, with a constant music genre, would the model learn the style of different artists ? And with the same instrument, would the model learn the difference between music genres ?

CONCLUSION & FUTURE WORK

The ability of conditional SampleRNN to learn classes is demonstrated when conditioned on speaker identity in voice datasets. Yet, it is obvious that the music-trained models lack consistency. It leads us to believe that combining the current architecture with hand-engineered features specific to music would lead to a much better generation. Since Music Transformer[12] produced such great results, combining the long-term

[10] https://soundcloud.com/sample-rnn/sets/full-voice-generation-epoch

[11] https://soundcloud.com/sample-rnn/sets/complete-batch-ambient

[12] https://magenta.tensorflow.org/music-transformer

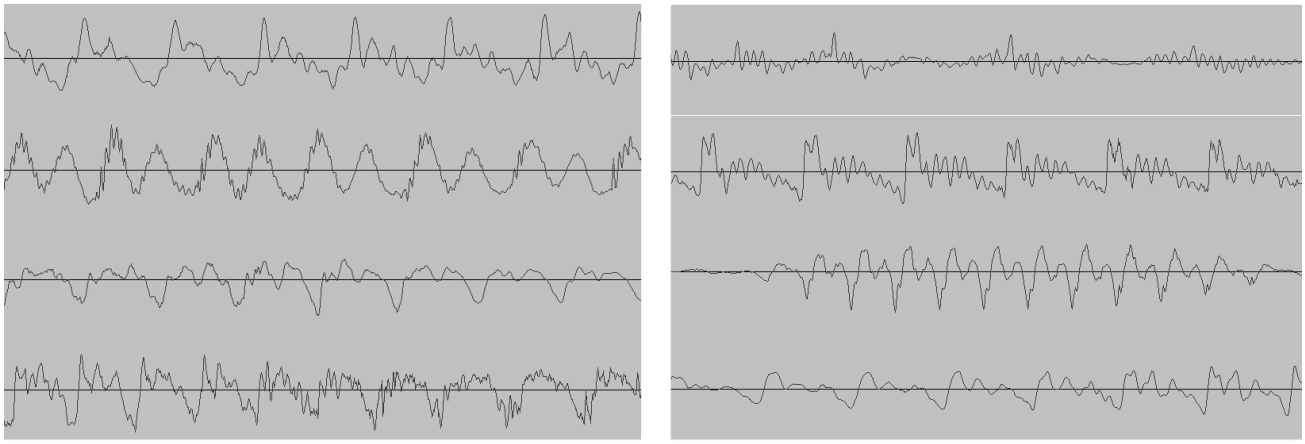

Figure 3. 50ms of audio. To the left, generated samples. To the right, training samples. From top to bottom, Paul Boehmer, Stephen Fry, Perdita Weeks and Thandie Newton. The structures are similar. The last generated sample looks noisy and it is indeed the worst one to the ear as well.

memory of the SampleRNN and relative attention mechanisms are bound to generate state-of-the-art samples.

REFERENCES

[1] David Alvarez-Melis CSAIL and Judith Amores. 2017. The Emotional GAN: Priming Adversarial Generation of Art with Emotion. *Nips* Nips (2017). https://nips2017creativity.github.io/doc/The

[2] David Huron, Neesha Anderson, and Daniel Shanahan. 2014. âĂIJYou Can't Play a Sad Song on the Banjo:âĂİ Acoustic Factors in the Judgment of Instrument Capacity to Convey Sadness. *Empirical Musicology Review* 9, 1 (2014), 29. DOI: http://dx.doi.org/10.18061/emr.v9i1.4085

[3] Nal Kalchbrenner, Erich Elsen, Karen Simonyan, Seb Noury, Norman Casagrande, Edward Lockhart, Florian Stimberg, Aaron van den Oord, Sander Dieleman, and Koray Kavukcuoglu. 2018. Efficient Neural Audio Synthesis. (2018). http://arxiv.org/abs/1802.08435

[4] Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron Courville, and Yoshua Bengio. 2016. SampleRNN: An Unconditional End-to-End Neural Audio Generation Model. (2016), 1–11. http://arxiv.org/abs/1612.07837

[5] Mohammad Soleymani, Anna Aljanaki, and Yi-hsuan Yang. 2018. DEAM : MediaEval Database for Emotional Analysis in Music. (2018), 2–4. http://cvml.unige.ch/databases/DEAM/manual.pdf

[6] Yading Song, Simon Dixon, and Marcus Pearce. 2012. Evaluation of musical features for emotion classification. *Proceedings of the 13th International Society for Music Information Retrieval Conference, ISMIR 2012* Ismir (2012), 523–528. https://pdfs.semanticscholar.org/351a/1e449d70ea4288320c2b8f4a74ddf14547be.pdf

[7] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. WaveNet: A Generative Model for Raw Audio. (2016), 1–15. http://arxiv.org/abs/1609.03499