Deep learning models for generating audio textures

Lonce Wyse and Muhammad Huzaifah*

National University of Singapore, Singapore
lonce.wyse@nus.edu.sg, e0029863@u.nus.edu

Abstract. Audio textures are a superset of standard musical instrument timbres that include more complex sounds such as rain, wind, rolling, or scraping. With appropriate modeling strategies, textures can be synthesized under parametric control analogous to the way musical instruments are, and can then become a powerful creativity tools for music making. However, audio textures, with complex structure spanning multiple time scales, are a challenge to model and generate synthetically. They are even challenging to define. Deep learning approaches offer new ways to develop generative audio texture models, and they create different demands on training data than traditional modeling approaches, In this paper we briefly review previous modeling approaches, and attempt to rationalize and converge on a definition of textures using modeling concepts. We introduce a new and growing data set along with a system for managing metadata specifically designed for audio textures. Finally, we report on some recent advances in texture models that are capable of generating sounds substantially beyond the range of sounds on which they are trained.

Keywords: audio texture, sound database, generative models, audio synthesis

1 Introduction

1.1 Modeling Objective

The overall goal of this work is to create generative models of audio textures, a class of sounds quite different, and arguable much larger and more complex, than the class of pitched musical instrument sounds, or even speech. Ideally, the models should be capable of convincingly generating "natural" textures such as rain, crowd murmur, crackling fire, or wind. They should offer control parameters that can be designed to correspond to arbitrary paths through the space of sounds within a model's range (for example controls that are perceptually or semantically meaningful). The models should be responsive to control parameters in real time, be capable of generating novel sounds between and beyond the

* This research is supported by a Singapore MOE Tier 2 grant, "Learning Generative Recurrent Neural Networks," and by an NVIDIA Corporation Academic Programs GPU grant.

sounds used as training examples, and finally, be able to generate sound for any length of time without repetition and without quoting the data on which they were trained.

1.2 Defining textures

The literature on audio texture modeling is full of discussions and definitions of that grapple with similar concepts, but a definition has proved elusive. Saint-Arnaud and Popat (Saint-Arnaud & Popat, 1995) plotted the relationship between time and the information content in three classes of sound: 1) noise, 2) texture, and 3) speech and music. As time progresses, the information in noise signals plateaus early, while for speech and music, information continues to rise with time. The graph for sound textures is in between that of the other two classes. It plateaus, but at a higher level than noise, and at a point further along the time access. This captures one of the key common concepts concerning sound textures, that there exists a window of time beyond which the description of the sound remains the same.

It is worth being a little more precise. If the generating model is not known ahead of time, and the model output never repeats itself exactly, then the curve representing the information content of the sound approaches an asymptote but does not ever reach a slope of zero. Similarly, when we train a generative model, each new piece of data leads to further refinement of the model.

Some authors have attempted to define textures by qualities inherent in the sound rather than how it is modeled. For example, Deimo Schwarz, who has made seminal contributions to complex and textural sound synthesis based on granular and corpus-based techniques (Schwarz, 2007), classifies contact sounds from interaction with objects, such as friction and rolling sounds, as not textures (Schwarz, 2011). This is because they violate the "wallpaper" property with the premise that fine structure must remain constant over time. Similarly, Strobl et al. (Strobl, Eckel, Rocchesso, & Le Grazie, 2006) rule out the sound of a crying baby as a texture because "the characteristics of the fine structure are not constant enough". These intuitions about how a perceptible "arrow of time" undermines the static nature of audio textures can be clarified in two different ways with the help of some model-based terminology.

One is to separate the description of the information that is not constant over shifting windows of time from the rest of the description. That is, to recognize a layer of "content" upon which the textural part is conditioned. For example, a rolling stone has weight and rate characteristics which determine characteristics of the resulting sound. If those characteristics change systematically over time, then so does the texture. The distinction between content and texture depends on what is being explicitly modeled across time, and what aspects of variation are drawn from a distribution that is constant across time. Thus the content/texture distinction is made in terms of the model used to generate, describe, or perceive a sound, and can be delineated differently even for one and the same audio example. Applying different listening strategies for a given sound is a topic of study in auditory psychology (Bregman, 1994) as well as in music

(Smalley, 1996). Another way to contextualize sounds with a strong "arrow of time" as textures is by not considering them in isolation, but *en masse*. For example, if a collection of rolling stones were heard, each having its own (possibly systematically changing) speed chosen from a random distribution, then the resulting sound would be a texture because the description would be constant over windows at different times. It is not that two different time windows onto the resulting sound would be indistinguishable, but rather that they are drawn from the same parameterization of the same distribution. If a person listening to this sound were to distinguish between two temporal windows of the audio, hearing and/or describing how they differ, they would not be listening to the sound as merely a texture. Again we see that the quality of "texture" belongs to a model (generative, descriptive, or perceptual), not to the sound.

1.3 Previous texture modeling strategies

Past approaches to synthesizing audio textures have used granular synthesis (Roads, 1978) incorporating a distinction between fine time scale of audio and the larger scale of grains, and wavelet trees that capture statistical relationships across both time and hierarchical scale (Dubnov, Bar-Joseph, El-Yaniv, Lischinski, & Werman, 2002) based on individual examples. McDermott and Simoncelli (McDermott & Simoncelli, 2011) developed a purely synthetic method starting from white noise and imposing specific statistics to iteratively adjust the system to match the paramaterization of statistics of natural sounds. This worked well on sounds with variation at shorter time scales, but less well when sounds had longer-term structure, such as footsteps.

Selecting specific time scales *a priori* to differentiate between "atoms" and audio makes artificial distinctions that do not always hold in natural sounds such as howling wind or friction. More recent deep learning approaches tend to avoid engineering features, and to let the model identify which features are important for a given data set.

2 PATSet - an audio texture database

Training neural networks typically requires large amounts of training data. There are good reference datasets for music (Bertin-Mahieux, Ellis, Whitman, & Lamere, 2011), musical instruments (Engel et al., 2017), and environmental scenes (Gemmeke et al., 2017), but they are either not specific to audio textures, or else labeled for training classifiers or unconditional generators, rather than for training synthesizers under parametric control (Huzaifah & Wyse, 2020). We initiated a new data base, Parameterized Audio Textures Data Sets (PATSets, available online[1]) collection to address this need.

The PATSet collection is small but growing, and consists of both natural and synthetic textures. Each set consists of multiple files, most often with each

[1] https://sonicthings.org:9999/

file corresponding to a single value of a parameter (e.g. engine speed) that can be used for conditional training. Individual files can also consist of different examples where the target parameter changes in time. An example is water filling containers where the "fill level" is the parameter, and out of practical necessity, is constantly changing. To handle the parameter management (labeling, writing, and reading files), the paramManager was developed, based on a JSON-like parameter file format.

The paramManager is an open source project on GitHub[2]. Any number of parameters can be stored for each sound file, and each can be changing over time. A key feature is that each parameter is stored as a pair of time and value arrays, each at a rate appropriate for their dynamic behavior. For example, they can be sampled at much lower rates than the audio files, and rates do not have to be regularly spaced in time. The paramManager code for writing parameter files is easily integrated into sound analysis or synthesis code, and the API for reading values at specific times during training interpolates between the possibly irregularly spaced time/value pairs.

Since neural networks are frequently trained with normalized parameter values, standard supplementary fields can be used to provide the natural units and values which often carry semantic value (e.g. MIDI note numbers, revolutions per minute) and can be retrieved for creating generative control interfaces in natural units.

The entry for each sound set in the PATSet database includes other metadata about how the sound was recorded or constructed as well as technical specifications (bitrate, channels, coding). It is possible to provide URLs or specify the source of synthetic sounds, as well as to upload the code (apps or source code) used to generate them. This is so that the integrity of the sound descriptions can be tested and verified, and so that sound sets other than than those already stored on the database can easily be generated. This can be handy for generating more or less densely sampled parameter values, or test sets that differ from training sets. Currently, the PATSet database allows auditioning of pregenerated stored sounds, and files are converted to the required sample rates when they are downloaded.

3 Experiments

We are currently generating and training models mostly with synthetic data sets in order to systematically explore the capabilities and limits of the models (testing long time dependencies in RNNs for example). We are exploring an RNN that was previously developed for modeling musical instrument tones (Wyse, 2018). An RNN is used because it is formally causal and real time where samples are produced in sequence and can be responsive to control parameter changes within the duration of one audio sample. This is in contrast to CNNs (and other architectures) that produce an extended duration of output for each parameter

[2] https://github.com/lonce/paramManager

vector update. The model consists of 3 layers of 300-800 GRU units each, takes input consisting of one dimension for mu-law encoded audio, and one dimension per parameter used for conditioning. The model has been shown to generalize well across a sparsely-trained pitch space, but struggles to generate timbres "in between" instruments used in training. Here we report on the ability of this architecture to model sound textures, far more complex than pitched instrument tones, with statistical variation across a continuous range of time scales.

To demonstrate the core competence of this model for modeling textures, we chose one of the synthetic sound sets in the PATSet database, "RegularPops68". The sound is constructed of a series of regularly spaced "pop" events at different rates with rate provided as the parameter for conditioning during training and control during synthesis. Each pop consists of 3 random samples of audio followed by a narrow band-pass filter with a center frequency of 415Hz (MIDI note 68). The 3 random samples give a significantly different timbre and amplitude to every single event. We expect the model to generate the constant variation at the fine temporal scale as well as the conditioned rate at the longer time scale. (see Fig.1, with corresponding audio online[3]).

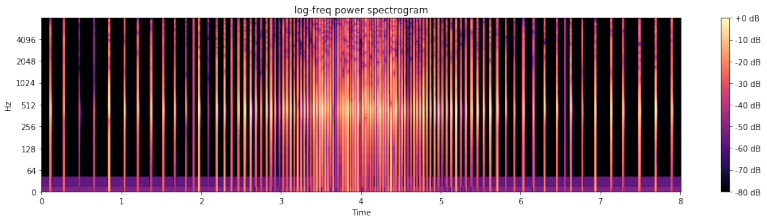

Fig. 1. Generated audio from an RNN trained on regularly spaced audio filtered random noise "pops" conditioned on the rate parameter. Both the constant variation of timbers and the overall filter shape are well captured and reproduced.

RNNs have well-recognized limitations on the length of the time dependencies they are able to model. Prior empirical studies have demonstrated that LSTM-type recurrent units and its derivatives like the GRU can retain information in its cell state for on the order of a thousand time steps (Hochreiter & Schmidhuber, 1997). At a 16kHz audio sampling rate, events spaced by 0.25s are already spaced at 4000 samples, and we do observe that as the event rate of the textures slows below 4 events/s, the generative model becomes less able generate the desired rate determined by the conditioning parameter. This is true for both randomly spaced and regularly spaced events. For this reason, we developed the Multi-tier Conditioning Recurrent Neural Network ("MTCRNN") with different tiers dedicated to modeling different times scales. A similar idea was previously explored in the musical domain for separating MIDI note generation from audio generation (Manzelli, Thakkar, Siahkamari, & Kulis, 2018).

[3] https://animatedsound.com/research/AICM2020

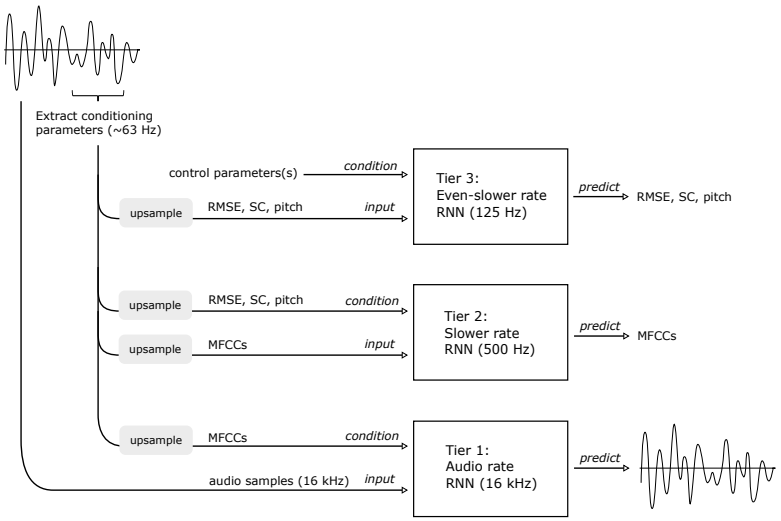

Fig. 2. Example of training an MTCRNN with 3 tiers. Audio is used to train tier 1, while intermediate representations at lower sample rates are used to train other tiers. The user controlled parameter conditions the highest tier only.

The multiple tiers of the MTCRNN are each stacked RNNs, and each are trained conditionally on different representations of the audio signal that have different sample rates corresponding to the tier. The representation of the conditioning input at each tier is the representation learned by higher tiers. Fig.2 shows an example of an MTCRNN with 3 tiers, with tier 1 training on audio at 16kHz. Root Mean Square Energy (RMSE), Spectral Centroid (SC), pitch, and Mel Frequency Ceptstral Coefficients (MFCCs) are extracted from the audio data at rates appropriate for the representation. Each representation is used at one tier as training data, and then at the subsequent lower tier as conditioning input for different data representations sampled at higher rates. The tiers are trained entirely independently. During the generative phase, the user conditions with parametric input only at the highest tier, while the conditioning input for lower tiers are generated in a cascading fashion, and then upsampled to match the sample rate and to condition generation at the next tier.

Two sounds from the PATSet datbase were used (PoissonGeiger and RegularPop), each exposing a "rate" parameter for controlling events per second (randomly and regularly spaced, respectively). Given that rate is a measurable quantity defined over a finite duration makes it a convenient way to probe time dependencies. To keep the rate (and pattern) stable, the model has to maintain some notion of a count of the sound events it has already produced over a trailing time window.

The MTCRNN was able to capture both the regularly spaced events in RegularPop and the random distribution of events from PoissonGeiger, as illustrated in Fig.3 and 4 respectively. This was true even with relatively large periods of silence between events, with the pop sound in Fig.3 spaced about 4000 samples apart at $r = 0.0$. This demonstrates the model's ability to maintain stable audio characteristics over significant time periods, leading to coherent long-form textures.

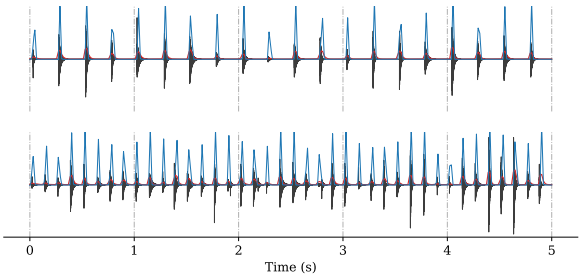

Fig. 3. Synthesised RegularPop waveform with $rate = 4 * 2^r, r = 0$, producing 4 events/s (top), and $r = 0.334$, producing about 8 events/s (bottom). The 2-tier MTCRNN model here conditions the sample-level tier with information pertaining to onset strength (blue) and RMSE (red).

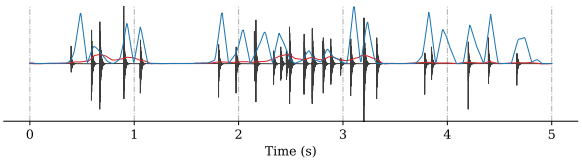

Fig. 4. Synthesised PoissonGeiger waveform with r = 0.5. The 2-tier MTCRNN model here conditions the sample-level tier with information pertaining to onset strength (blue) and RMSE (red).

A comparison between a 1-tier model and a 2-tier MTCRNN also shows the benefits of separate training at different time scales. The second tier of the 2-tier model was conditioned on rate to synthesise RMSE and onset strength, which were in turn used to condition the audio tier, while the 1-tier model generating audio was conditioned directly on rate. The number of pop events counted over a 10s duration is shown for both 1 and 2-tier models in Fig.5. It is evident that the 2-tier model is much more accurate than the 1-tier model in terms of

reproducing the correct time dependencies according to the specified rate. The 1-tier model tended to generate overall fewer events than desired according to the rate parameter, and was more prone to producing irregular pops. Long time dependencies are captured better with the multi-tier model.

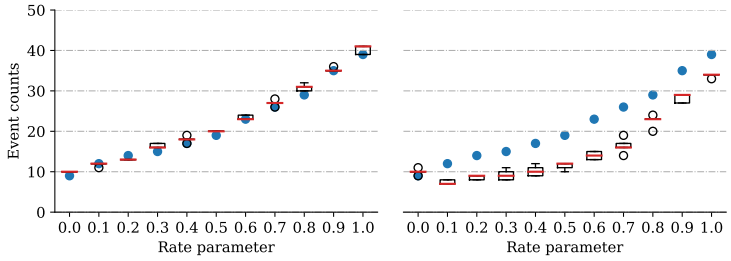

Fig. 5. Boxplots of the total number of events generated over 10s at various rates by a 2-tier model (left), and 1-tier model (right), trained on the RegularPop dataset. Each boxplot is a distribution over 5 different synthesised instances. The event counts from the real data is shown as a blue dot.

Fig.3 also illustrates generalization to produce events at parameter values not in the training set (which were spaced at .1 in [0,1]). Moreover, the model was found to be surprisingly good at extrapolation beyond the trained rates. Tests showed that the correct underlying rate was maintained up to a parameter rate value of around 3, triple the maximum value of the parameter used in training. The ability to direct the audio output on such a flexible scale with a comparatively sparse amount of data opens up many avenues for the creative use of the system for generating novel audio.

4 Summary

We are developing tools for generating complex audio textures with structured information across a broad range of time scales. The concept of texture is elusive without reference to the representation of a sound. RNNs provide freedom for the creative use of this interesting class of sounds in audio design and music contexts by allowing the user to specify which aspects of a sound will be controlled over time as "content" and which aspects will be drawn from a learned conditional statistical distribution as "texture". The classical single-tier RNN is limited in its ability to model the long time dependencies which means that the window of time over which a texture can be defined is limited. To address this issue, we are creating a sound database specifically designed for exploring the representational capabilities of RNNs. We also developed the multi-scale MTCRNN for improving time scale modeling capabilities and for interactively generating novel audio material for musical purposes.

References

Bertin-Mahieux, T., Ellis, D. P., Whitman, B., & Lamere, P. (2011). The million song dataset. In *Proceedings of the 12th international society for music information retrieval conference (ismir 2011).* Miami, FL, USA.

Bregman, A. (1994). *Auditory scene analysis: The perceptual organization of sound.* Boston, Massachusetts: MIT Press.

Dubnov, S., Bar-Joseph, Z., El-Yaniv, R., Lischinski, D., & Werman, M. (2002). Synthesizing sound textures through wavelet tree learning. *IEEE Computer Graphics and Applications*, *23*(4), 38–48.

Engel, J., Resnick, C., Roberts, A., Dieleman, S., Norouzi, M., Eck, D., & Simonyan, K. (2017). Neural audio synthesis of musical notes with wavenet autoencoders. In *International conference on machine learning* (pp. 1068–1077).

Gemmeke, J., Ellis, D., Freedman, D., Jansen, A., Lawrence, W., Moore, R., ... Ritter, M. (2017). Audio set: An ontology and human-labeled dataset for audio events. In *2017 ieee international conference on acoustics, speech and signal processing (icassp)* (pp. 776–780).

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Huzaifah, M., & Wyse, L. (2020). Deep generative models for musical audio synthesis. *arXiv preprint arXiv:2006.06426*.

Manzelli, R., Thakkar, V., Siahkamari, A., & Kulis, B. (2018). Conditioning deep generative raw audio models for structured automatic music. *arXiv preprint arXiv:1806.09905*.

McDermott, J. H., & Simoncelli, E. P. (2011). Sound texture perception via statistics of the auditory periphery: evidence from sound synthesis. *Neuron*, *71*(5), 926–940.

Roads, C. (1978). Automated granular synthesis of sound. *Computer Music Journal*, *2*(2).

Saint-Arnaud, N., & Popat, K. (1995). Analysis and synthesis of sound textures. In H. G. Okuno & D. Rosenthal (Eds.), *Readings in computational auditory scene analysis.* Erlbaum.

Schwarz, D. (2007). Corpus-based concatenative synthesis. *IEEE Signal Processing Magazine*, *24*(2), 92–104.

Schwarz, D. (2011). State of the art in sound texture synthesis. In *14th int. conf. digital audio effects.* Paris, France.

Smalley, D. (1996). The listening imagination: Listening in the electroacoustic era. *Contemporary Music Review*, *13*(2), 772–107.

Strobl, G., Eckel, G., Rocchesso, D., & Le Grazie, S. (2006). Sound texture modeling: A survey. In *Proceedings of the 2006 sound and music computing (smc) international conference.* Marseille, France.

Wyse, L. (2018). Real-valued parametric conditioning of an RNN for interactive sound synthesis. In *6th international workshop on musical metacreation (arxiv preprint arxiv:1805.10808).* Salamanca, Spain.