Using Recurrent Neural Networks to Dream Sequences of Audio

Andrew Pfalz
Louisiana State University
apfalz@lsu.edu

Edgar Berdahl
Louisiana State University
edgarberdahl@lsu.edu

Jesse Allison
Louisiana State University
jtallison@lsu.edu

ABSTRACT

Four approaches for generating novel sequences of audio using Recurrent Neural Networks (RNNs) are presented. The RNN models are trained to predict either vectors of audio samples, magnitude spectrum windows, or single audio samples at a time. In a process akin to dreaming, a model generates new audio based on what it has learned.

Three criteria are considered for assessing the quality of the predictions made by each of the four different approaches. To demonstrate the use of the technology in creating music, two musical works are composed that use generated sequences of audio as source material.

Each approach affords the user different amounts of control over the content of generated audio. A wide range of types of outputs were generated. They range from sustained, slowly evolving textures to sequences of notes that do not occur in the input data.

1. INTRODUCTION

A number of related projects have already used recurrent neural networks (RNNs) for predicting audio waveforms. This paper begins by reviewing some of these projects before explaining how a different approach is taken, with the goal of being able to dream of audio waveforms that may be helpful in music composition applications.

1.1 Related Work

The Deep Dream project has shown that a convolutional neural network can be used to create images that are not in a given data set. The model rearranges and combines images it has seen in surprising ways, similar to collage. Similarly, Roberts et al. [1] applied the Deep Dream algorithm directly to audio waveforms. The results from those experiments are not appropriate for the types of applications this work is concerned with. The process of generating sounds using such an algorithm involves deducing through an iterative process what information is stored in the various layers of the model. The data being sonified is not the output of the model, but rather a way of listening to what the model learned. As such, there is no way using such an algorithm to get an output prediction based on a specified input.

An alternative approach for generating novel sequences of audio is to have Long Short Term Memory networks (LSTMs) predict them. Related work demonstrates how to use LSTMs to generate novel text and handwriting sequences [2]. WaveNet introduce an approach for modeling sequences of audio waveform samples [3]. SampleRNN shows impressive results for modeling sequences of audio in less time [4].

In the field of music, using machine learning and more specifically LSTM architectures has been explored. The Google research group Magneta [5] demonstrates the generation of sequences of musical notes using symbolic notation. Similarly, Nayebi and Vitelli [6] attempt to predict audio using GRU cells. Kalingeri and Grandhe [7] investigate this issue as well using various different architectures. Some of the most thorough investigations in this field have been carried out by Cope [8].

The char-rnn project demonstrates predicting text one character at a time with either recurrent neural networks and LSTMs. It produces novel sequences after being trained on an input corpus [9]. The models produce output that features unexpected reordering of the input data. The combinations of words the models produce are often quite realistic and may appear verbatim in the input data. Other times, the sequences are rather strange and make little sense.

1.2 Goals

The goals of the present work are to find an architecture that can reliably produce output audio sequences that:

1. are relatively free from artifacts,
2. obviously resemble the input, and
3. do not merely reproduce the input exactly.

One of the interests of the authors is to apply this technology as a digital audio effect, where the user has a reasonable expectation of the quality and content of the output so that the results can eventually be used in a music composition.

1.3 Background on Neural Networks for Prediction

Neural networks are a technology for training computers to perform complex tasks. The training procedure begins with presenting some input to the model. The model produces some output in response. The fitness of this output is evaluated via a loss function. For the present work, the loss function in (1) is the mean squared error between the label s Y_i and predictions from the model $\hat{Y}_i$. This loss function is appropriate because it encourages the RNNs to generate predictions to match the labels, and the loss function is equivalent to finding the loss in the frequency domain.

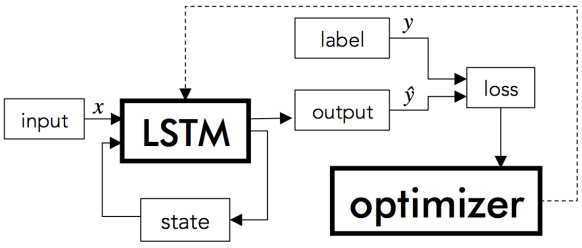

Figure 1. How an optimizer can be used to train an LSTM for prediction tasks. During training, the model produces an output prediction based on the input data and updates the state, which stores short term information about recently seen inputs. The fitness of this output is calculated via the loss function. The optimizer teaches the model to make better predictions by updating the internal parameters of the model.

$$loss = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2. \quad (1)$$

This loss value is passed to the optimzer, which updates the internal parameters of the model via backpropogation. By iteratively repeating this process the model gradually learns to perform the task. Ideally neural networks learn a generalized fixed mapping from inputs to outputs.

When predicting sequential data, it can be beneficial for the neural network to be able to model short-term dependencies. If a model is trying to predict the next word in a sentence, it needs to know general information about language like that adjectives often precede nouns in English. It also needs to know information about the local context of the the word it is trying to predict. For instance, it is helpful to know what the subject of the previous few sentences is. RNNs were developed to solve this challenge by incorporating internal state variables to help provide the network with memory.

Long Short Term Memory networks (LSTMs) are a specific kind of RNN. As shown in Figure 1, they have a mechanism, called the state, for storing and recalling information about the data the network has seen. Hence, if the model had seen several sentences that mention France and it is trying to predict the next word in the sentence "I can speak," it could use the short term memory stored in its state to guess that "French" would be an appropriate word. [1] In contrast with non-recurrent neural networks, LSTMs can learn both long and short term dependencies, so just as LSTMs can work well for predicting character sequences, LSTMs can work well for predicting complex sequences of audio samples.

2. PREDICTING SEQUENCES OF AUDIO

2.1 Overview

Given an LSTM that has been trained to predict sequences of audio from some specific training data, the authors employ a similar autoregressive algorithm for generating audio sequences. First a seed of length `n` is chosen either from the training dataset or from some other audio source. Then, based on this seed and in the same way as it would during training (compare Figures 1 and 2), the model makes its first prediction, which also has length `n`. The final element of the predicted output (the element at index `n-1`) is then appended to the end of seed as shown in Figure 2. The first element from the previous seed is then discarded so that the next seed has the same length `n`.

The seed thus acts as a first in first out queue (see Figure 2). The LSTM predicts what comes after the seed, then guesses what comes after its first prediction and so on as the total output sequence is generated.

[1] `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`

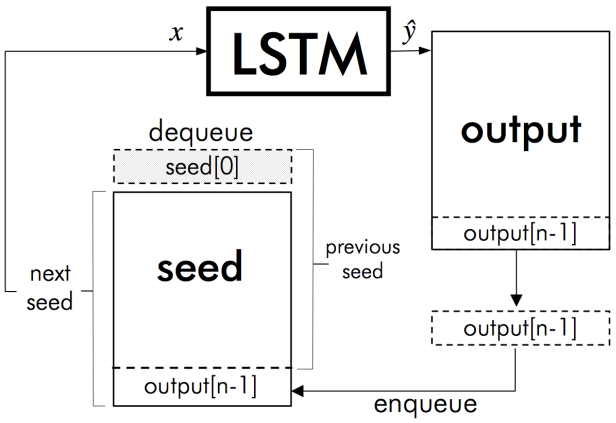

Figure 2. How the authors use the LSTM model to predict sequences of audio. After the initial seed has been shown to the model, it makes a prediction about the following audio in the same manner as during training. To generate an output sequence the final element from the prediction is appended to the seed and first element from the seed is discarded. This process repeats until the total desired output length is reached.

2.2 Analogy to Dreaming

When the model makes predictions about data it has never seen, it sometimes makes small errors. During training the optimizer corrects the model when these errors occur. During generation though, there is no evaluation of the loss function. We use the analogy of dreaming here to mean that using this process, the model can output sequences that may be based in a real input but have no fixed restriction. This resembles the same way that a dream often begins with a normal setting and then becomes increasingly less realistic—the model predictions begin with the seed, but then are free to move on to more and more surreal sounds.

2.3 Four Prediction Approaches Investigated

Four different approaches are investigated for specifically formatting the input and output data for the training and prediction structures (see Figures 1 and 2). The differences between these approaches are important as they cause some differences in the subjective quality of the predictions.

2.3.1 Audio Vector Prediction

For the audio vector approach, the input to the model at each iteration is a series of non-overlapping audio vectors of length 1024 samples. If the input vectors are indexed from `0` to `n`, the output of the model is trying to match is audio vectors indexed from `1` to `n+1`. During generation, the final vector from the output is appended to the end of the seed, and the first vector in the seed is removed.

2.3.2 Magnitude Spectrum Prediction

For the magnitude spectrum approach, the input and output data are represented in the frequency domain. Specifically, the input data is comprised of the normalized magnitude spectra of audio waveforms. The negative frequencies are are omitted and the entire dataset is normalized to the range `0` to `1` by dividing every bin by the maximum value in all of the frames. Consequently, the phase information from the Fourier transform is discarded.

During training and generation, the data is presented to the model in the same manner as with the audio vector approach. After a sequence of predictions has been generated, the output audio is re-synthesized using a channel vocoder, implemented with an oscillator bank. Since the phase information from the original audio is not present in the input or predictions, the vocoder artificially advances the phases of the oscillators.

Some noise is introduced into the audio by way of the behavior of the prediction scheme shown in Figure 2. For the magnitude spectrum prediction, a simple technique is applied to slightly attenuate this noise: the resynthesis algorithm finds frequency bins with magnitudes falling beneath a target threshold and sets these magnitude values to 0 to silence them [10].

2.3.3 Transpose Approach

The input data is formatted the same for this approach as for the audio vector approach. The difference is that the output of the model is a single sample, rather than a vector of audio. This is accomplished by passing the output of the LSTM to a fully connected neural network layer with one output neuron. The resulting output is a column vector, the transpose of which is passed to a second fully connected neural network layer with a single output neuron, resulting in a single scalar value. During the generation phase, the seed is formatted in the same way as in the audio vector approach. The audio sample that is predicted by the model is appended to the end of the seed and the first sample of the seed is removed (see Figure 2).

2.3.4 Column Vector Prediction

With this approach, the input data is presented to the model in the same format as with the audio vector approach. The difference is that the model predicts a single audio sample for each input vector. For example, if an input vector contains audio samples indexed from 0 to 1024, the job of a model making predictions with this approach would be to make a prediction of audio sample 1025. To produce an output, only the final sample in the column vector is kept and appended to the end of the seed.

3. EVALUATION

3.1 Evaluation Criteria

Evaluation of the sequences generated in these experiments is not completely straightforward, because there is no obvious quantifiable metric to use to rate audio sequences that a neural network is dreaming. We therefore present three criteria to evaluate the success of the generated outputs.

Criterion 1 is that the outputs should be relatively artifact free. By artifact we mean any aspect of the generated sequence of audio that is undesirable for use in music, especially aspects that are difficult to remove via filtering or other means [11, 12]. For an output sequence to score low on this criterion, the artifacts present in the audio must be so pervasive or so difficult to overcome that the audio is unusable in musical applications.

Criterion 2 is that the dreamed audio should obviously resemble the training samples. By this we mean that a reasonable listener can easily discern some similarities in frequency and timbre between the training audio and the dreamed audio. For example, if the input for the model were piano recordings, two cases of failure for this criterion would be pink noise or a sine wave as the output. In the case of noisy outputs, these would score poorly on criterion 1 as well. They also fail criterion 2 because most reasonable listeners would say that noise does not resemble piano. Likewise a sine wave in the output would fail to accomplish criterion 2 because, even though there are no apparent artifacts in the prediction, it does not resemble the training audio any more than it does any other musical input. Dreamed audio that does not resemble the input is penalized because those sounds could easily be produced using an oscillator or noise generator.

Criterion 3 is that the dreamed audio should not resemble some portion of the training audio so closely that the two cannot easily be distinguished once any artifacts in the generated audio are discounted. One example would be if the model outputs several seconds of the training audio unchanged, or with minor artifacts introduced such as short passages of low amplitude noise. In this case the model clearly learned to output the correct answers when presented with specific training input, but did not learn to generate any new data or to generalize from what it learned and apply this information in new circumstances. Scoring high on criterion 3 would entail in the least reordering of the pitches from the input or some other significant transformation of the data that cannot obviously be repeated through some other means. (This is similar in spirit to a computational novelty measure [13].)

3.2 Methodology

As the four prediction approaches were investigated and tested using the structures in Figures 1 and 2, hundreds of audio sequences were generated and evaluated, while the settings of the model (henceforth hyperparameters) were gradually adjusted. The hyperparameters that proved most significant were the learning rate, the standard deviation of the noise the weights were initialized to, and the length of the input vectors. The speed at which new audio sequences could be generated and the machine learning setup were limited by the fact that LSTMs generally require a lot of computational power for training and testing. Accordingly, the authors have kept notes on how the different prediction approaches performed in a wide variety but not completely exhaustive set of examples.

The work presented here is in its preliminary stages. The focus of these experiments is on creating a reliable and consistent model that affords the user a reasonable degree of control over the content of the outputs. Questions of whether the model would generalize well to specific other audio sources was not a concern, and has not yet been investigated in general. The intention of these experiments

is to arrive at a model or series of models that are maximally consistent in producing outputs that are easy to use in the style of musical compositions that suit the authors' aesthetic goals. The next section reports on the general patterns that the authors observed while aiming to get the prediction approaches working. All of the results presented here were created using input data sampled at 11025Hz.

3.3 Sampling Methods

Two approaches for generating outputs are investigated. In the first algorithm, training is paused periodically and the model produces an output before resuming training. In the second algorithm, the model generates outputs concurrently with training. In practice, many of the pitfalls of the first approach are alleviated by using the second approach. All of the audio examples presented here were produced using the concurrent approach.

3.4 Results

The general results are described below and are also highlighted in Table 1.

The audio vector approach has the benefit of being the fastest method for producing audio outputs. The outputs generated with this method tend to feature windowing artifacts, however. They are related to the length of the audio vectors being predicted. Changing the length of the audio vectors has a dramatic effect on the quality of the outputs

The magnitude spectrum approach is more successful than the audio vector approach. The outputs generally feature fewer undesirable artifacts. The model produces a wide range of results that succeeded in accomplishing criteria 3. Like the audio vector approach, changing the hop size of the FFT used to produce the magnitude spectra significantly changes the quality of the outputs.

The dreamed outputs from the transpose approach reliably score highly on all three criteria. They hardly ever feature noise, and when it is present it is easily overcome through filtering. The dreamed outputs rarely feature passages where criteria 1 or 2 are not met. The dreamed outputs from this approach feature sustained pitches from the training audio with subtle changes in timbre throughout. This approach is much slower than the preceding two because the many more iterations of prediction are required to produce an output of the same length.

The column vector approach produces outputs that most closely resemble what humans might produce if presented with the same task as the model. When presented with a series of pitches, for example, a human would likely predict another series of pitches. The other approaches would sometimes predict textures that contrasted sharply with the prevalent texture in the input data. The column vector approach is also slow because only a single sample is kept at each iteration.

4. AUDIO EXAMPLES

We present a few audio examples from each of the generation approaches. Relatively successful outputs were chosen for each approach. All the outputs were produced with the same audio input. The extent to which each audio example meets or does not meet the evaluation criteria are summarized in Table 1. The audio files themselves are available to stream along with the input data at `https://apfalz.github.io/rnn/rnn_demo.html`

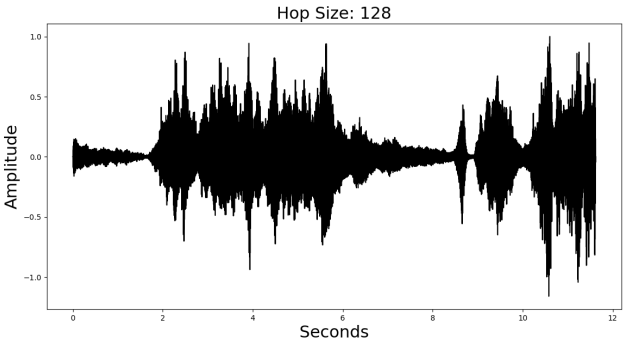

Figure 3. Audio example 8 shows that the structure from Figure 2 is able to dream audio waveforms of significant complexity. To listen to more generated audio waveforms, please visit the project webpage at `https://apfalz.github.io/rnn/rnn_demo.html`

4.1 Audio Vector Approach Examples

The most significant factor that effects the content of the outputs produced with this approach is the length of the audio vectors being predicted. **Audio example 1** was produced by asking the model to predict audio vectors of length 1024. The audio features a sustained texture with some noticeable swells, but for the most part individual attacks are not discernible. The output audio resembles concatenative or granular synthesis.

As the length of the audio vectors being predicted is decreased (see **Audio Examples 2-4**), more clarity in terms of pitch events begins to emerge. However, for the most part the outputs produced with this approach lack clearly defined attacks. Instead, notes and timbres from the input data swell irregularly, sometimes overlapping each other.

4.2 Magnitude Spectrum Approach Examples

When a large hop size is chosen, the outputs produced tend to feature sustained pitches from the input superimposed on each other. Hardly anything that resembles an attack can be heard. **Audio Example 5** is an example of this situation, which uses a hop size of 1024.

As the hop size is decreased (see **Audio Examples 6-8** and Figure 3), more clarity is present in the sampled outputs. **Audio Example 9** for instance contains a series of notes from the input data. A warbling effect can be heard later in the sample where the model went unstable and predicted a series of contrasting magnitude spectra.

4.3 Transpose Approach Examples

Audio outputs produced with this approach afford the user the most control over the frequency content of the outputs. This comes at the expense of any clearly defined attacks. These outputs consistently feature all of the frequency content of the seed in a sustained texture. **Audio Example 10** is an example of this. To produce an output with a specific set of pitches, the user only needs to construct a seed that contains only the desired pitches.

	Criterion 1 (Artifacts)	Criterion 2 (Resembles Input)	Criterion 3 (Does not Match Input)
Audio Example 1 (Audio vector approach, length 1024)	medium	high	high
Audio Example 2 (Audio vector approach, length 512)	medium	high	high
Audio Example 3 (Audio vector approach, length 256)	medium	high	high
Audio Example 4 (Audio vector approach, length 128)	medium	medium	high
Audio Example 5 (Magn. spect. appr., hop size 1024)	high	medium	high
Audio Example 6 (Magn. spect. appr., hop size 512)	high	medium	high
Audio Example 7 (Magn. spect. appr., hop size 256)	high	high	high
Audio Example 8 (Magn. spect. appr., hop size 128)	high	high	medium
Audio Example 9 (Magn. spect. appr., hop size 56)	medium	high	low
Audio Example 10 (Transpose approach)	medium	high	high
Audio Example 11 (Column vector approach)	medium	high	high

Table 1. Summary of the extent to which the audio examples are meeting or not meeting the evaluation criteria. A score of low in a criterion indicates the output completely fails or nearly completely fails to meet that criterion. A score of medium indicates the output achieved relatively high but does not totally meet the criterion. A score of high indicates the output either totally met or nearly totally met the criterion.

4.4 Column Vector Approach

This approach produces outputs that most closely resemble what a human would predict were he or she presented with the same task as the model. In **Audio Example 11** separate notes are clearly heard with intervening intervals of silence. The sequence of notes the model predicted does not occur in the input data. When the data is presented to the model in this format, the model appears to learn the most about how to predict new audio. At times the model does produce outputs with this method that distort the input data in surreal ways. The most common occurrence of this is when notes that should decay are sustained unnaturally. Having never heard a guitar note, for instance, that sustains for multiple seconds the model nonetheless predicts just that.

5. MUSICAL APPLICATIONS

We present two original compositions that demonstrate how the sounds created by an LSTM can be used.

5.1 Like two strings speaking in sympathy

The piece *Like two strings speaking in sympathy* is a composition for guitar with live processing. The goal for this piece was to use sounds generated using the sample approach in a live setting. The process of generating outputs using the transpose approach is far too slow to be run in real time during a performance. In order to use the sounds in a manner that is more interactive, outputs were generated ahead of time. The live input from the player is analyzed in real time by a Max/MSP patch. It listens to the player and chooses which of the previously generated sounds to accompany the player with.

The dataset that the LSTM was trained on to create the sounds contains the same pitches as the live guitar part. To create separate outputs with specific pitch content, seeds were created that contained only some desired pitches. They were shown to the LSTM, which was asked to generate an output sequence. A subset of the generated sounds are further processed in real time in Max/MSP. They are subjected to ring modulation and an envelope is applied to them.

5.2 What comes is better than what came

What comes is better than what came is another composition for guitar with live electronics. It is meant to showcase the magnitude spectrum approach. In this case, an LSTM was trained on a similar dataset to that for *Like two strings speaking in sympathy*. The model was then shown seeds from the melody played by the guitar. The predictions have the rough outline of the melody, but feature the pitches from the training dataset.

In a sense the experience of the model, having only ever heard darker more dissonant audio, predisposes the model to predict similarly dissonant audio when presented with a new input. During the performance, the sequences can be generated in near real time and played back in a staggered manner, or they can be generated ahead of time and played back by a trigger sent to Max/MSP.

6. CONCLUSIONS

We presented four approaches for generating audio from an LSTM. These methods allowed the model to consider a large amount of input data and to create new outputs that are based on the the input. The model predicted either vectors of audio, magnitude spectrum frames, or single samples at a time. To generate an output, the model made predictions based on its own prior predictions iteratively until the desired output length was reached.

We intended for the sequences predicted by the model to be usable in musical compositions. In order to evaluate how successful the models were in creating viable outputs, we presented three criteria. To be score highly, an output needed to be relatively free from artifacts, to obviously resemble the input, and to not merely reproduce the input exactly.

Our primary concern was narrowly defined to be reliability and control. For this technology to be viable for use in music composition, users must be able to produce a large number of audio sequences that fall within a predictable range of possibilities. The various approaches presented here afford the user a range of levels of control over the content and quality of the audio samples.

Each approach for generating outputs presented here has strengths and weaknesses. The audio vector approach has

the benefit of requiring relatively little time to compute, and matching the timbre of the input very closely. It tends to feature artifacts that are difficult to overcome. The magnitude spectrum approach has the benefit of affording the user control over the clarity of the pitch events in the generated output, and also being relatively quick to compute. The outputs generated with this approach tend to not match the timbre of the input very closely though. The transpose approach has the benefit of being very consistent and reliable and affording the user very precise control over the pitch content of the outputs. But the outputs tend to feature only sustained sounds and they require a relatively large amount of time to compute. The column vector approach produces outputs that score the highest on criteria 2 and 3, but the they tend to feature low amplitude noise and they are also slow to compute.

7. REFERENCES

[1] A. Roberts, C. Resnick, D. Ardila, and D. Eck, "Audio Deepdream: Optimizing raw audio with convolutional networks." 2016.

[2] A. Graves, "Generating Sequences With Recurrent Neural Networks," *CoRR*, vol. abs/1308.0850, 2013. [Online]. Available: http://arxiv.org/abs/1308.0850

[3] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, "WaveNet: A Generative Model for Raw Audio," *CoRR*, vol. abs/1609.03499, 2016. [Online]. Available: http://arxiv.org/abs/1609.03499

[4] S. Mehri, K. Kumar, I. Gulrajani, R. Kumar, S. Jain, J. Sotelo, A. C. Courville, and Y. Bengio, "SampleRNN: An Unconditional End-to-End Neural Audio Generation Model," *CoRR*, vol. abs/1612.07837, 2016. [Online]. Available: http://arxiv.org/abs/1612.07837

[5] E. Waite, "Generating Long-Term Structure in Songs and Stories," https://magenta.tensorflow.org/2016/07/15/lookback-rnn-attention-rnn/, 2016.

[6] A. Nayebi and M. Vitelli, "GRUV: Algorithmic Music Generation using Recurrent Neural Networks," 2015.

[7] V. Kalingeri and S. Grandhe, "Music Generation with Deep Learning," *CoRR*, vol. abs/1612.04928, 2016. [Online]. Available: http://arxiv.org/abs/1612.04928

[8] D. Cope, *Virtual music: Computer synthesis of musical style*. MIT press, 2004.

[9] A. Karpathy, "The Unreasonable Effectiveness of Recurrent Neural Networks," http://karpathy.github.io/2015/05/21/rnn-effectiveness/, 2015.

[10] D. Wang, U. Kjems, M. S. Pedersen, J. B. Boldt, and T. Lunner, "Speech intelligibility in background noise with ideal binary time-frequency masking," *The Journal of the Acoustical Society of America*, vol. 125, no. 4, pp. 2336–2347, 2009.

[11] B. Smith, "Instantaneous companding of quantized signals," *Bell Labs Technical Journal*, vol. 36, no. 3, pp. 653–709, 1957.

[12] R. Dolby, "An audio noise reduction system," *Journal of the audio engineering society*, vol. 15, no. 4, pp. 383–388, 1967.

[13] J. Foote, "Automatic audio segmentation using a measure of audio novelty," in *Multimedia and Expo, 2000. ICME 2000. 2000 IEEE International Conference on*, vol. 1. IEEE, 2000, pp. 452–455.