# LATENT BIRDS: A BIRD'S-EYE VIEW EXPLORATION OF THE LATENT SPACE

**Juan Alonso Moreno**
Aalborg University Copenhagen
jalons19@student.aau.dk

**Francesco Bigoni**
Aalborg University Copenhagen
fbigon17@student.aau.dk

**George Palamas**
Aalborg University Copenhagen
gpa@create.aau.dk

## ABSTRACT

The use of a generative approach for sound synthesis breaks through the limitations of traditional approaches, proposing novel ways to explore creative ideas. This paper demonstrates a method to generate original bird vocalizations using a Variational Convolutional Autoencoder trained on mel-spectrograms of bird song and call recordings. The vocalizations are reconstructed by sampling the latent space and decompressing the resulting mel-spectrogram. The results are quite promising, in that our system is able to generate a variety of bird vocalizations depicting plausible songs and calls, by interpolating between existing vocalizations or sampling the latent space. A Twitter bot that publishes a unique daily bird vocalization is also implemented.

## 1. INTRODUCTION

The last decade has led to the emergence of a new generation of artists, exploring the possibilities of humans and machines as equal collaborators. Although at first the creative applications of machine learning (ML) were limited, mainly due to the lack of creative workflows, the insufficient volume of data and the inherent complexities of utilizing an ML model, in recent years a range of tools and frameworks have become available, bringing this discipline closer to the public.

In our study, we used Keras [1] to explore the creative potential of Neural Networks (NN) for sound synthesis, with an easy to train architecture.

Our motivation stems from the following needs:

1. There are few architectures capable of synthesising or generating natural sounds. Systems such as [1] require a complex setup and can generate only very short audio fragments (1 second). A workflow capable of generating almost unlimited longer bird vocalizations can be used as a copyright-free system for adding ever changing sound effects. This includes, but is not limited to, the film and gaming industries, television and radio broadcasts, artistic installations, wellness and mindfulness purposes.

2. Having a publicly available collection of bird vocalizations (distributed by a Twitter bot) and available, open-source tools, would enable practitioners and enthusiasts of ML and creative programming to test their methods, compare them and encourage novel research in the field.

Inspired by the *Bird Sounds* project by Manny Tan and Kyle McDonald [2], where the authors use t-SNE [3] to organize thousands of bird sounds in a two-dimensional grid, we decided to employ existing bird vocalizations to train a Variational Convolutional Autoencoder (VCAE) to generate new bird vocalizations. The generated audio reaches the public through a Twitter bot implemented specifically for this purpose; the Jupyter Notebook containing our implementation is available on GitHub.

## 2. BACKGROUND

### 2.1 Neural Networks and Audio

When it comes to sound and music generation, ML-based audio processing is still broadening its horizon: the main areas of research are speech synthesis [4] and music and sound generation [5].

The use of deep neural networks to create longer output sequences on a sample-by-sample basis is especially relevant in this context. The two most common architectures are *WaveNet*, based on convolutional layers [6], more stable but slower to converge, and *SampleRNN*, based on recurrent layers and different time steps [7] that converges much faster, but sometimes induces unstable performance.

These technological advances are also used by the musicians as new tools, built entirely from scratch, or using frameworks such as Google Magenta, whose mission is defined as "exploring the role of machine learning as a tool in the creative process" [2]. The outcome of this technological democratization is projects such as Audio Deepdream [5] where the NN is used to hallucinate soundscapes, or the work by Carr and Zukowski [8], that feed a Recurrent NN with instances of several music styles in order to compose full albums. Generative Adversarial Networks (GAN) have been explored in [1], where the authors implement and compare two GAN architectures, the first one based on raw audio and the second one based on spectrograms, with no

[1] https://keras.io/

[2] https://magenta.tensorflow.org/

conclusive results in terms of human preference for one of those two methods.

### 2.2 Spectrograms and Convolutional Networks

By converting the audio into a graphical representation, image processing techniques can be applied. The spectrogram is the most commonly used graphical representation, because it retains information from both time and frequency domains. This representation can be further improved by applying a non-linear transformation that mimics the behaviour of the human ear by reducing the resolution of the higher frequencies. The resulting spectrogram (called mel-spectrogram [9]) is fed into a Convolutional Neural Network (CNN), where a series of filters or kernels are created and applied to all the input pixels, to extract low- and high-level features. Then, a pooling operation is used to reduce the size of these features. This pair of operations is usually repeated several times until reaching a desired level of image abstraction. The CNNs are a popular choice for a range of sound and music classification and detection applications, but they are not widely used for audio generation.

### 2.3 Variational Autoencoders

Since raw audio files are large, a lot of processing power is needed to handle them in a reasonable amount of time. In this context, dimension reduction is a tool for limiting complexity and efficiently explore a multidimensional space of possible new instances. The Autoencoder [10] is a type of NN architecture divided in two parts (Figure 1):

1. The *encoder* compresses the information: each layer has a smaller dimension than the previous one, until reaching a bottleneck layer, which holds the most dense representation of the data, efficiently performing dimension reduction;
2. The *decoder* restores the original data, using the code vector as its input.

As input and the output have the same dimensions, this system can be seen as a lossy compression scheme.

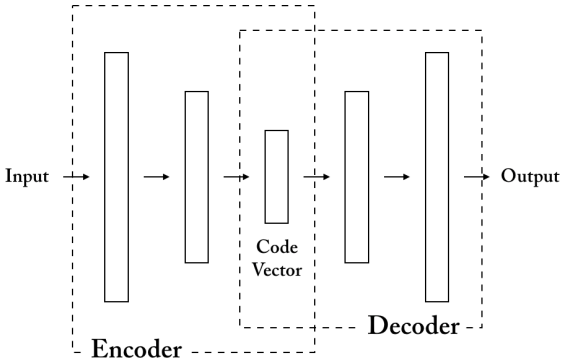


Figure 1. Architecture of a deep autoencoder. It consists of an input and an output layer of the same dimensions, and a number of hidden layers.

The intuition is that, once the autoencoder is trained, new outputs can be generated by varying the code vector, an operation often referred to as "sampling the latent space". Also, interpolation between different inputs is possible by interpolating the code vectors generated by these inputs. The main drawback is that, depending on the original data distribution, the obtained latent space generated is usually not smooth: thus, moving from point $a$ to point $b$ in the latent space does not guarantee a continuous and meaningful series of outputs. Furthermore, the smaller the code vector dimension, the more irregular the latent space will be. To avoid this pitfall and ensure a smoother latent space, a new generative model has been proposed: the so-called Variational Autoencoder (VAE) [11].

In a Variational Autoencoder, the encoder generates a stochastic space made of normal distributions $N(\mu, \sigma)$ rather than points. Then, the network is forced to learn a more relaxed representation of the input. For this architecture, the loss function is rewritten as the sum of two terms: the reconstruction loss (e.g. mean square error) applied on the output, and the regularization loss applied on the code vector to force a standard normal distribution [11] (i.e. $\mu = 0$ and $\sigma = 1$). Figure 2 depicts the additional layers $\mu$ and $\sigma$ that encode the mean and the standard deviation of every element of the code vector.

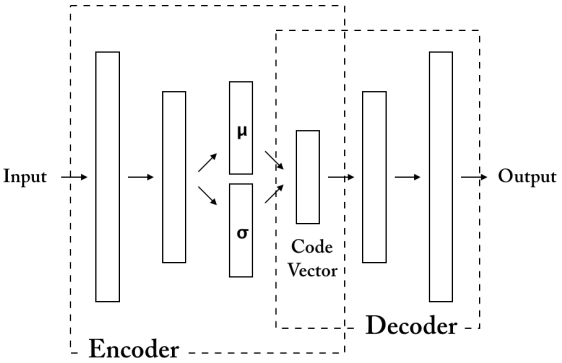


Figure 2. Representation of a Variational Autoencoder. The extra layers regularize the latent space, forcing a more compact grouping of the encoded values, thus avoiding overfitting and allowing better results during the generative process.

### 2.4 Ornithology and Machine learning

Birds use two kinds of vocalizations: songs, associated to territory and courtship, and calls, associated to keeping contact among the flock and raising alarms. The distinction depends on context, length and complexity of the vocalizations. In the field of ornithology, the study of bird songs is highly regarded, as birds are one of the few biological classes that share extensive vocal learning demands with humans: "birds need to hear and imitate others in order to develop their vocal communication signals" [3] . These skills are absent in the greater apes, more closely related to hu-

[3] https://gtr.ukri.org/projects?ref=BB%2FR008736%2F2

mans than birds. So, modeling the birds' learning process can help us understand how humans learn to communicate.

There is also a growing interest in bird detection using only sound. In the first Bird Audio Detection challenge [12], the winning team [13] used an implementation based on CNNs. In BirdCLEF 2018, a convolutional architecture is also used for setting the baseline score [14].

Despite the efforts being made in speech or instrument sound synthesis, no special work is being done on the synthesis of bird vocalizations, with some exceptions based on 1) Hidden Markov Models [15] based on human speech but taking into account the fluctuations of the vocalizations and the unpredictability of the utterances or 2) particle swarm simulations [16], where bird songs are synthesized by modulating an oscillator by means of Bezier curves computed by a particle swarm.

## 3. DESIGN AND IMPLEMENTATION

Our implementation is made with Keras and the TensorFlow [17] backend in a Jupyter Notebook.

### 3.1 Data

Our dataset is extracted from *Cornell Guide to Bird Sounds: Essential Set for North America* [4] and contains 1379 files, for a total of six hours and two minutes of audio. These files have been cut into four-second fragments and downsampled to 22050Hz (16-bit, mono). To augment our data, we used an overlap of 2 seconds on the original instances. We discarded any audio clip shorter than 4 seconds.

The audio fragments were converted into 128 bins $\times$ 128 frames mel-spectrograms using *librosa* [18] with an FFT window length of 4096 samples (185.7 ms), a hop size of 690 samples (31.3 ms), Hamming windowing and 128 mel-bands. Note that phase information was thereby discarded. We converted our spectrogram values to a dB scale and stored as 16384-element vectors.

### 3.2 Training

The resulting set of spectrograms was used to train a VCAE (Figure 3). The **encoder** stage has the following architecture:

- Input layer: 16384 neurons, one for each pixel in the spectrogram.
- A Reshape layer, to transform the one dimensional input into a $(128, 128)$ array.
- Six Conv2D (i.e. convolutional) + MaxPooling2D (i.e. pooling) layers for stacking the filters and progressively reducing dimensionality.
- One Flatten layer to get a 1024-dimensional vector.
- Three 512-neuron Output layers, i.e. two Dense layers in parallel, which store $\mu$ and $\sigma$ (the parameters of each distribution), and one custom layer (Lambda) for retrieving samples from the latent space using the values from the two Dense layers.

[4] https://store.birds.cornell.edu/product_p/ml-essential-1.htm

The six Conv2D layers stack the convolution filters (32, 64, 128, 256, 512 and 1024, respectively) using a $3 \times 3$ kernel size, with a stride of $(1, 1)$. The MaxPooling2D layers reduce the dimensionality from $128 \times 128$ to $64 \times 64$, $32 \times 32$, $16 \times 16$, $8 \times 8$, $4 \times 4$, and $1 \times 1$. All the layers use a pool size of $(2, 2)$ except the last one, which is $(4, 4)$.

The variational layers are based on the Keras variational autoencoder example [5] , including the *reparametrization trick* to compute the derivative needed for backpropagation through stochastic nodes.

The **decoder** is a network of four fully connected layers, which progressively increase the number of neurons from 512 to 16384 (512, 2048, 4096, 16384 respectively) to reconstruct a $128 \times 128$ spectrogram.

We use LeakyReLU [19] as the activation function in all layers except the last one, where we use a sigmoid function, and in the stochastic layers, where we use the default linear activation function.

Due to the extreme differences among bird vocalizations, the full augmented dataset is shuffled and split 80-20 into training and test sets (7256 and 1815 spectrograms respectively) so fragments from the same vocalization will be randomly assigned to one set, including partially overlapping fragments. The NN is trained for 1500 epochs with a batch size of 128, using Adam [20] as the optimizer with a learning rate of 0.00025. We use a custom composite loss function with two terms: the standard mean error squared, plus the regularization loss.

The encoder and decoder are defined as separate models, but we also define a model that encompasses them, so that the user can access the encoder, the decoder or both at the same time.

The generated output is reshaped to a $128 \times 128$ array to recompose the mel-spectrogram; then, the frequency bin amplitudes are converted from dB to linear scale; finally, librosa's function mel_to_audio (which first approximates an STFT magnitude from a mel-spectrogram and then performs an approximate spectrogram inversion using the Griffin-Lim algorithm [21]) is invoked to reconstruct the waveform.

## 4. RESULTS

The VCAE training took 4 hours on a Titan X. At the end of the process, we got a training loss of 58.47 and a validation loss of 192.27 (both unitless), as shown in Figure 4. We chose to overfit the VCAE as the main use of this model is exploring the latent space.

After training, we used the model in two different ways:

- **Vocalization generation.** To generate a new vocalization, we sample the latent space, generating a $512 \times 1$ vector, where each value is sampled from a normal distribution. This vector is fed to the trained decoder, that outputs a spectrogram corresponding to a new (i.e. non previously existing) vocalization. We can also move in one (or several) dimensions of the latent space, generating a series of evolving bird

[5] https://keras.io/examples/variational_autoencoder/

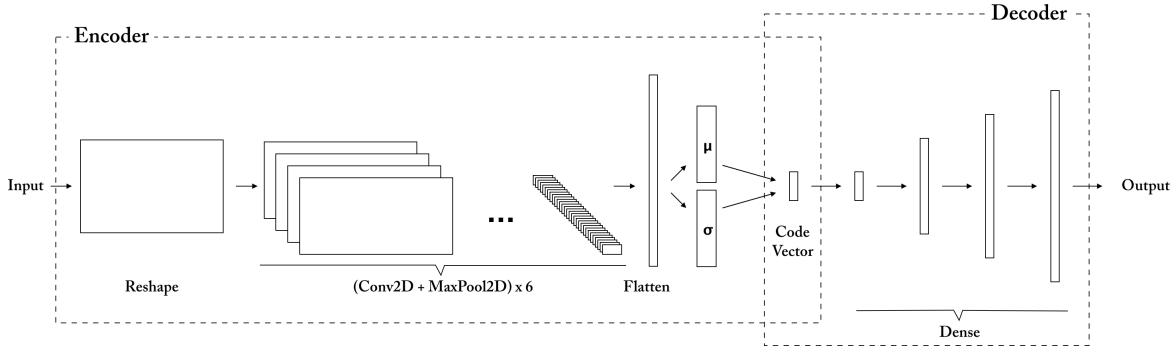


Figure 3. Variational Convolutional Autoencoder Architecture used in this paper. The input spectrogram is reshaped and presented to six convolutional + pooling layers and then flattened to estimate a distribution in the latent space. These distribution is sampled and reconstructed using a deep decoder.

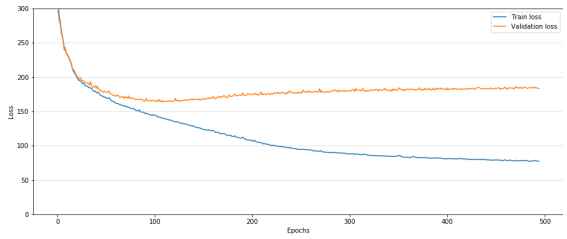


Figure 4. Mean square error loss over 500 training epochs for train and validation sets.

vocalizations. The spectrogram is finally converted to audio.

- **Vocalization interpolation.** To interpolate between bird vocalizations, we randomly choose two spectrograms $s_1, s_2$ from the training set, run them through the encoder and get two coded representations of the vocalizations $r_1, r_2$ (i.e. two different distributions in the 512-dimensional latent space). The interpolated code vector $r_i$ is calculated using this formula: $r_i = \alpha \cdot r_1 + (1-\alpha) \cdot r_2$, where $0 \leq \alpha \leq 1$. $r_i$ is then fed to the decoder to generate a spectrogram which is finally converted to audio. This transformation in shown in Figure 5.

Our system is able of generating 14000 spectrograms per second. Each spectrogram produces 4 seconds of audio, and the conversion takes 0.75 seconds, so, if required, a continuous stream of audio in real time can be produced.

300 files have been generated as content for the Twitter bot (150 new vocalizations, 150 interpolated vocalizations). For each vocalization, a still picture has been generated, showing an overlay of the spectrogram and the waveform.

## 5. DISCUSSION

To evaluate the robustness of the VCAE architecture, we made a series of tests:

- We plotted the representation of two randomly chosen dimensions of the mean vector, and checked that the values conform to the expected distribution.
- We plotted the activation of the filters in the first convolutional layer. The results matched our expectations, as the plot highlighted different features of the spectrograms (edges, thresholds...)
- We presented it a fragment from the training set and compared its reconstruction with the original audio using both the spectrogram and the audio file. The reconstructed spectrogram and audio file were highly similar to the original files.

Figures regarding these checks are available in the GitHub repository.

A pure convolutional architecture without the variational layer would provide a more compact code vector, and produce a smaller validation loss, but for the purpose of a generative model, smoothness of the latent space is more desirable than a perfect reconstruction.

By using the VCAE architecture, we introduce an extra layer of complexity. In exchange, the latent space gets reorganized in such a way that when building a random code vector or interpolating two vocalizations, the results are more bird-like.

Other types of architectures have been tested (e.g. symmetrical deep autoencoder and symmetrical convolutional autoencoder) with worse performance than our non symmetrical convolutional encoder/deep decoder.

The generated audio demonstrated some interesting characteristics. The number of unusable vocalizations (noisy output, garbled sounds) is low. A higher amount of vocalizations exhibit what we could call a *flocking effect*. This effect is due to the energy spread in the reconstructed spectrograms: the energy bands in the generated spectrogram tend to lose definition and get spread into a wider frequency range. At the same time, some artifacts appear (see Figure 6). These two factors translate perceptually into the sound of a flock of birds rather than a single one, as is perceived as several birds singing not at the same exact frequency, as in a real flock. When the system generates

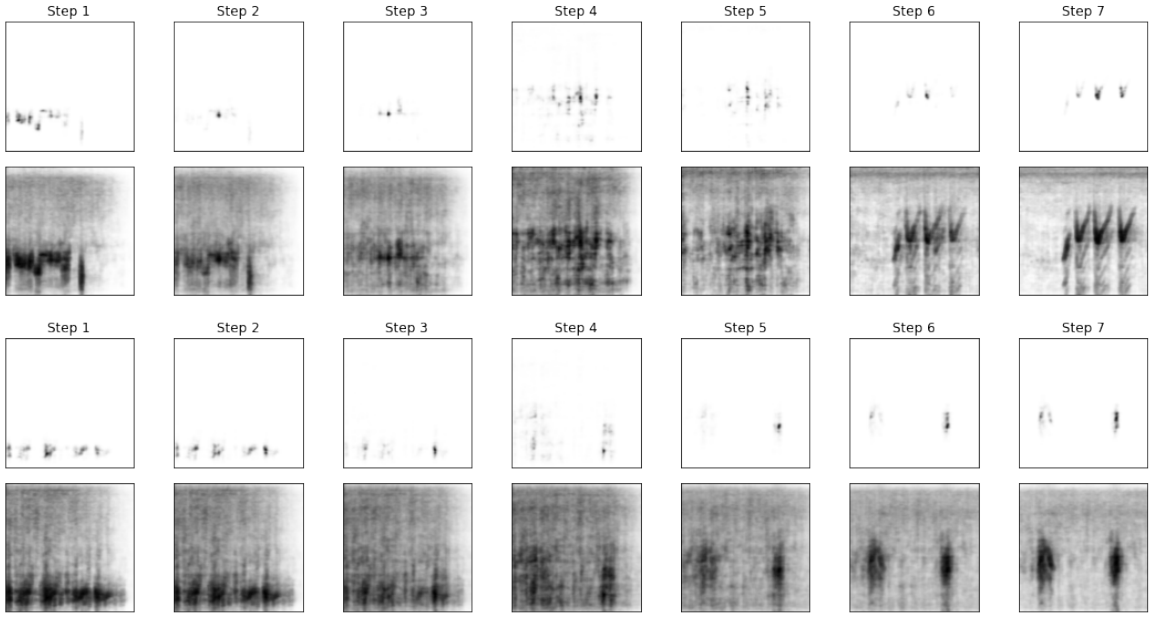


Figure 5. Seven-step interpolation examples. Between fragments 3507 and 6215 (Top two rows) and fragments 6058 and 683 (Bottom two rows). Rows 1 and 3 show the power mel-spectrogram. Rows 2 and 4 show the dB mel-spectrogram). As can be observed, the intermediate steps in the latent space are more complex than merely crossfading between spectrograms.

a spectrogram with narrow high-energy bands, the flocking effect disappears, and the vocalization is perceived as if sung by a single bird.

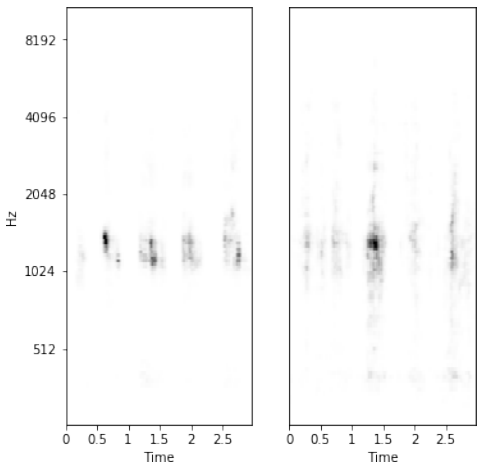


Figure 6. Distribution of energy in an original spectrogram (left) and in the reconstructed version (right) showing the spread of the energy and the flocking effect.

By using a 512-dimensional code vector, the latent space becomes less compact than desirable. This can be observed at the midpoint of some interpolations, where the generated audio is garbled and noisy.

To obtain longer audios using the proposed architecture, we start by choosing an original sample and calculating the corresponding coordinates in the latent space. By orbiting around that point we can obtain similar sounds. To create a richer soundtrack, orbits with different radii and on different dimensions can be combined. The smaller the radii, the more similar the vocalizations will be to the original audio. By adding Perlin noise to the orbit radius, we can generate a continuous audio stream without wandering away from the original sample.

## 6. CONCLUSIONS

In this work, we have investigated bird vocalization synthesis. For that, we have designed and implemented a nonsymetrical Variational Convolutional Autoencoder capable of generating 4-second bird vocalizations in real time. Two vocalization generation methods are used: random sampling of the latent space and interpolation between existing vocalizations. The recreated sounds present interesting characteristics, proving that the approach presented is a valid step towards the synthesis of nature sounds.

The proposed architecture is very fast to train compared to other alternatives (e.g. GAN-based methods), taking hours instead of days; it generates longer vocalizations; finally, it is relatively simple to extend or integrate in existing workflows.

Our code is available on GitHub [6], including the pretrained model and the spectrograms, but without the original audio files. Also, a Twitter bot [7] is posting a new bird vocalization every day.

[6] https://github.com/juanalonso/latentbirds
[7] https://twitter.com/latentbirds/

## 7. FUTURE WORK

Our results are promising, but there is room for improvement. More specifically:

- Although a large variety of sounds have been used as a training data-set (tweets, hoots, chirps, woodpecking, etc.), some of them were underrepresented, rendering specific sound qualities more dominant to the generative procedure. Using the full Cornell database, or getting new vocalizations from other datasets (e.g. xeno-canto [8]) could improve the representation of the missing types of vocalizations. Also, applying existing knowledge about the structure of bird vocalizations [12] can improve the preprocessing of the audio and the architecture of the VCAE.
- An interesting possibility is feeding the autoencoder with non-bird related spectrograms (music, speech, mechanical noises, other animal sounds) and see how the VCAE performs while trying to approximate those sounds.
- Use a labeled dataset to group birds of the same species and implement vector arithmetic to interpolate between them.
- Improve the latent space by discriminating the less interesting/unusable bird vocalizations. This can be achieved by exploring the latent space checking for the existence of outliers and analyzing the presence of common characteristics or by creating an additional architecture to evaluate the quality of the dataset.

### Author contributions

Alonso is the main researcher and writer. Bigoni helped writing, editing and reviewing the manuscript. Palamas supervised the work and helped writing, editing and reviewing the manuscript.

## 8. REFERENCES


[1] C. Donahue, J. McAuley, and M. Puckette, "Adversarial audio synthesis," *arXiv preprint arXiv:1802.04208*, 2018.

[2] M. Tan and K. McDonald, "Bird sounds | experiments with google." [Online]. Available: https://experiments.withgoogle.com/bird-sounds

[3] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[4] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan *et al.*, "Natural tts synthesis by conditioning wavenet on mel spectrogram predictions," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4779–4783.

[5] D. Ardila, C. Resnick, A. Roberts, and D. Eck, "Audio deepdream: Optimizing raw audio with convolutional networks," in *Proceedings of the International Society for Music Information Retrieval Conference, New York, NY, USA*, 2016, pp. 7–11.

[6] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[7] 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," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017. [Online]. Available: https://openreview.net/forum?id=SkxKPDv5xl

[8] C. Carr and Z. Zukowski, "Generating albums with samplernn to imitate metal, rock, and punk bands," *arXiv preprint arXiv:1811.06633*, 2018.

[9] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE transactions on acoustics, speech, and signal processing*, vol. 28, no. 4, pp. 357–366, 1980.

[10] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[11] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *stat*, vol. 1050, p. 1, 2014.

[12] D. Stowell, M. D. Wood, H. Pamuła, Y. Stylianou, and H. Glotin, "Automatic acoustic detection of birds through deep learning: the first bird audio detection challenge," *Methods in Ecology and Evolution*, vol. 10, no. 3, pp. 368–380, 2019.

[13] T. Grill and J. Schlüter, "Two convolutional neural networks for bird detection in audio signals," *2017 25th European Signal Processing Conference (EUSIPCO)*, pp. 1764–1768, 2017.

[14] S. Kahl, T. Wilhelm-Stein, H. Klinck, D. Kowerko, and M. Eibl, "Recognizing birds from sound-the 2018 birdclef baseline system," *arXiv preprint arXiv:1804.07177*, 2018.

[15] J. Bonada, R. Lachlan, and M. Blaauw, "Bird song synthesis based on hidden markov models." in *INTERSPEECH*, 2016, pp. 2582–2586.

[16] J. Zúñiga and J. D. Reiss, "Realistic procedural sound synthesis of bird song using particle swarm optimization," in *Audio Engineering Society Convention 147*. Audio Engineering Society, 2019.


[8] https://www.xeno-canto.org/

[17] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 265–283.

[18] B. McFee, C. Raffel, D. Liang, D. Ellis, M. Mcvicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proc. of the 14th Python in Science Conf. (scipy 2015)*, 01 2015, pp. 18–24.

[19] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, vol. 30, no. 1, 2013, p. 3.

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[21] D. Griffin and J. Lim, "Signal estimation from modified short-time fourier transform," in *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 2. IEEE, 1984, pp. 236–243.