Autocoder: a Variational Autoencoder for Spectral Synthesis

David Brynjar Franzson
Intelligent Instrument Lab
Iceland University of the Arts
david.brynjar@gmail.com

Victor Shepardsson
Intelligent Instrument Lab
Iceland University of the Arts
victor@lhi.is

Thor Magnusson
Intelligent Instrument Lab
Iceland University of the Arts
thor.magnusson@lhi.is

ABSTRACT

We introduce the Autocoder, a simple machine learning tool for spectral synthesis and sound manipulation. The Autocoder is a creative framework that takes a sound — harmonic or inharmonic, monophonic or polyphonic — learns its surface features such as harmony, pitch and timbre, and generatively synthesizes a continuous soundscape in real-time, based on the trained model.

1. INTRODUCTION

A significant amount of energy has gone into developing generalized synthesis models based on large corpora of real–world sounds. These models often sacrifice local detail in order to capture the vast differences found in these large sets of data, making these models less than optimal as tools for producing case-specific high-quality sound output.

Retraining on larger datasets is also time and power expensive, making these models less useful for rapid creative experimentation. These tools mostly focus on monophonic sounds with harmonic spectra which limits their usefulness as a DSP tool. This paper positions a case specific creative tool — the Autocoder — as an applied solution.

The Autocoder [1] is a Tensorflow based implementation of a variational autoencoder that is optimized for singular musical input rather than large corpora of sounds, and attempts to model the spectral details of the input in order to synthesize a novel output based on the input sound.

It is available as a Max/MSP external, as a python framework, and as hardware in the form of a Eurorack module. It is designed to have high usability for non–expert users and allow even the most novice musician to experiment creatively with the underlying methods without a deep understanding of the topic.

1.1 Plain Autoencoders

An autoencoder [2] is a neural network that takes an input, runs it through one or more hidden layers and reproduces the input as accurately as it can. It can be described as a same-in-same-out structure where a compressed representation of the training data is learnt by the model.

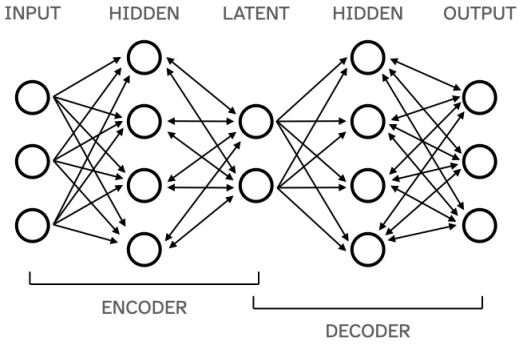

Figure 1. Plain autoencoder layout.

The network consists of two separate parts: an encoder that takes the training data and produces a latent vector encoding the training data in a lower number of dimensions; and a decoder that reconstructs the original input based on the latent vector (see Figure 1). The two parts are trained at the same time, using the reconstruction error, i.e. the difference between the input and the output, to adjust the weights of the hidden layers and in the process changing how the input maps onto the latent vector.

1.2 Variational Autoencoders

The representation learned by an unregularized autoencoder is not necessarily meaningful. If the model has no incentive to place similar data points near each other in the latent space, it may still achieve a low reconstruction error with a more-or-less arbitrary and contorted mapping into the latent space, making it 'lumpy'. As a result, minor changes in the latent space can produce wild variations in the output of the decoder, rendering it hard to control as a generative tool – there is no meaningful notion of a 'random point' in latent space, and a small perturbation can move off the data manifold entirely. The only reliable way to get a meaningful point in the latent space of a plain autoencoder is to run data through the encoder.

Variational autoencoders (VAE) [3] solve this problem by constraining the distribution of data points once embedded in the latent space. While the distribution in the latent space may be arbitrarily 'lumpy' when using an unregularized autoencoder, the VAE can require it to be in some sense 'smooth'; for example, to be Gaussian. The VAE achieves this by adding noise in the latent space during training, which encourages like points in the latent space to map to like outputs in the data space, while points in the latent space spread apart to become less confusable.

A complementary view is that the VAE approximates a

Bayesian probabilistic model of the data. With a VAE, we can sample points in the latent space from our chosen *prior distribution*. By passing those latent points through the decoder, they become samples from the (modeled) data distribution. As long as a vector in latent space is not too unlikely under the prior (i.e., has a magnitude less than three or so when using a standard normal prior), it should decode to something 'data-like', regardless of whether that specific latent representation is present in the training data.

1.3 Prior Work

The use of autoencoders for generative spectral synthesis was proposed by Sarroff et al. in 2014 [4]. They trained an autoencoder on a large corpus of magnitude Fourier frames and used the encoder to encode an incoming signal and manipulate the resulting latent vector before running the altered latent vector through the decoder, synthesizing an output based on the altered vector.

The basic autoencoder approach was further extended with MFCC transforms and made fully generative by Colonel et al. [5], training an autoencoder on a large corpus of samples taken from a MicroKorg synthesizer.

In their 2018 paper, Esling et al. [6] made a fully generative network, using a variational autoencoder with a mel scale transform of the input data, using Ircam's SOL database of monophonic instrumental sounds as training data. By manipulating any of the 64 values in the latent layer, hybrid instruments and perceptual descriptors were synthesized.

The Acids group at Ircam has developed this work further with their impressive RAVE model which allows for real-time unconditional generation and remapping of sounds between domains, with the output being generated directly from the model without the need for spectral resynthesis.[7]

The differentiable digital signal processing (DDSP) [8] project has made complementary progress on sound modeling by using an intermediate representation in the decoder between latent space and data space. With DDSP, a primary decoder outputs interpretable parameters for one of several differentiable synthesizers, which then complete the decoding to audio.

This prior work has in large part relied on larger scale corpora of monophonic sounds with harmonic spectra in order to produce a generalized model rather than modeling individual sounds. As a result, these models behave more like traditional instruments rather than an exploratory creative space and take days to weeks to train. In addition, much of the more recent work, such as Ircam's RAVE, focuses on the production of sequential waveform generation rather than the spectral domain.

2. MOTIVATION

Our motivation for the Autocoder is the artistic need for a simple tool aimed at creative individuals that allows for the manipulation and exploration of any input sound — harmonic or in-harmonic, monophonic or polyphonic — in real-time, with minimal time wasted on training and adjustment of network parameters while emphasizing the quality of the synthesized output. We found that existing technical solutions did not support the artistic outcome desired, and believe that the technical solution presented here will add knowledge and technical tools to the research domain of AI musical creativity.

3. IMPLEMENTATION

The Autocoder implements a VAE using the Tensorflow machine learning framework. The Autocoder uses a short input—such as a song or a song fragment—to optimize the latent dimension of the VAE based on similarities that are present in the specific input rather than optimize for all eventualities in a larger corpus. As the goal of the Autocoder is to generate materials that could exist within the space defined by the training data, the model can use the full input sound as both training and test input, which produces a more nuanced case specific output with shorter training times than when splitting the data into seperate training and testing input.

Random walks within the resulting latent space produce seemingly meaningful musical output based on features present in the input. This allows the artist to creatively explore and extend previously constructed materials rather than having to positivistically construct the musical experience from explicit parameters.

3.1 Training–data Preparation

A moment–by–moment spectral representation is extracted from an input sound by running it through a short-term Fourier transform. Any sequential or temporal structure that is not encoded within an FFT frame is lost. Any pitch connections that are present within a frame, either through harmony or when the frame overlaps over two consecutive sounds, are captured by the model. Through trial and error, we chose a frame size of 16384 in order to capture some of this local structure in the training data.

Raw FFT data has the issue that half of the data represents the top octave of the sound (which at a sample rate of 44.1kHz would be the frequencies between 11.025 kHz and 22.05 kHz), and half of the rest of the data the octave below that. Without specifying which frequencies are important to our hearing, the neural network would spend most of its capacity on modeling high-frequency details which are imperceptible to humans, neglecting more important spectral content. To address this, the amplitude of the spectrum is converted with a mel scale transform, a linear transform of the log distribution of energy in the spectrum. In the mel transform, each octave is represented by roughly the same number of values, making the reconstruction error more perceptually relevant.

Converting the data to mel also allows for a large compression of the input spectrum, in our case from 8192 points for an FFT window size of 16384 down to 512 points. The data is normalized by scaling the global minimum and maximum of the data between 0 and 1, and then fed into the encoder network.

The spectral amplitudes are absolute values, while the phase can be rotated and cuts off as it crosses either pi or -pi, making both raw phase and phase–difference hard to train on without some clever cooking of the raw data, so the phase of the input is ignored. This means that the synthesized output loses most transient information beyond that which is present in the larger scale dynamics of the input.

Future work will explore phase reconstruction techniques as a potential solution to this limitation.

3.2 Network Architecture

The objective of the network is to learn the spectral representation of the input data as accurately as possible in order to synthesize an output that is representative of the spectra and dynamics of the input data. The Autocoder architecture is somewhat arbitrary and designed through trial and error. It is a 'shallow' single hidden layer model that learns general features efficiently and fast. We found that using deeper architectures with our small-data tasks only resulted in a more 'lumpy' latent space (see 1.2).

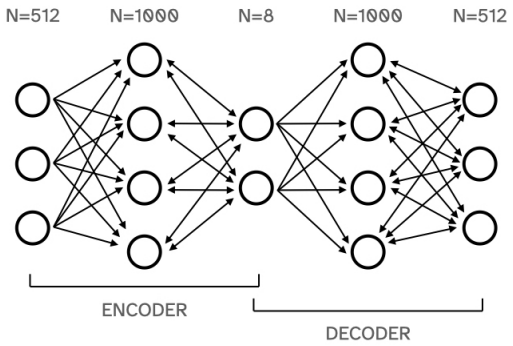

Figure 2. Shallow network architecture.

The hidden layer of the encoder explodes the input data of 512 points into 1000 dimensions, allowing for better feature separation, feeding into a latent space with 8 dimensions. The decoder reverses this process (see Figure 2). By feeding the training data through the encoder after training, min and max factors are extracted to scale the range of values in the latent vector that correspond to the min and max values in the encoded training data to the range from 0 to 1.

The model is trained using an Adam optimizer, with a Kullback Leibler loss function. The stopping condition is a given minimum change in the reconstruction error between epochs. Based on listening experiments the minimum change was defined as .001 for fast training, .0001 for medium amount of training, and .00001 for high amount of training. The learning rate was also defined as .001 for fast training, .0001 for medium amount of training, and .00001 for high amount of training. [1] As the learning rate and stopping conditions are adjustable, the number of epochs is variable. The resulting model size is roughly 6.4MB.

As the network architecture is relatively simple, training times on Google Colab are roughly equal to the duration of the input sound for a model using the fast training setting with batch size set to 256, and around 3-5x the duration of the input for a more detailed model. With batch size set to 4096, the training time of a model trained with the fast training setting drops to less than 25 percent of the original duration on a few minutes long input.

[1] Higher amount of training does not universally lead to better sounding results.

4. APPLICATIONS

Max/MSP implementations for 4.1 and 4.2 are included with the Autocoder[1]. A simple implementation of 4.3 can be found in the github repository.

4.1 Generative Synthesis

A continuous soundscape is produced by generating a random walk for each value in the latent vector and then decoding the latent vector into a spectral frame. The resulting spectrum is then synthesized by randomizing the phase and taking the inverse FFT of the frame. If the step size of the random walk produces undesirable discontinuities in the output, spectral domain low pass filtering can be applied.

Three examples of the output of the generative synthesis can be found at https://tinyurl.com/ya9jkz4d. In the first example, the shallow model was trained on a three minute song, in the second, the model was trained on three minutes from Wagner's Tristan and Isolde, and in the third, on 95 minutes from Books 1 and 2 of Monteverdi's Madrigals. [2]

4.2 Filtering

The network can also convert a discrete input space into a continuous one. Spectral filters can be derived from a set of reverb impulses which can then be fed as training data to the model, producing a continuous latent space that can be used to generate the filtering stage in a hybrid reverb. This generates new unheard impulses in-between the original impulses, and offers the ability to dynamically morph between discrete reverb spaces.

Models used for generative synthesis can also be used for convolution, resulting in a cross synthesis where the carrier signal is generated by the model and dynamic within the space defined by the training data. By whitening the modulator spectrum (by normalizing the frame to 0 to 1 and then setting each value to a fractional power, thereby flattening any peaks in the spectrum), the need for loud bins to align for a signal to be produced can be mitigated and a more forgiving imprint of the modulator on the carrier can be produced.

Two examples of the hybrid reverb/convolution can be found at https://tinyurl.com/2p97zmrw. The first uses a model trained on 660 impulse responses for the filtering stage of a hybrid reverb, while the second uses the generative output from the pop–song model from above as an impulse response.

4.3 Autocoding

A more direct form of cross synthesis can be produced by feeding a different input sound into a pre–trained encoder, producing a latent representation of that new input sound as 'heard' by the AI. This representation can then be decoded, effectively transferring timbre from the model onto the input sound. Since the AI only knows how to hear things based on its training data, a hybrid sound is produced. It is highly unlikely that the latent vector representations even as much as overlap, and since the model is trained to fit very tightly to the input data with minimal generalization, the two sounds need to be trained together

[2] The Autocoder can handle longer input as long as it is internally consistent and structured.

as a single model, and the latent representation of the modulator must be offset, clipped and scaled to match the latent representation of the carrier. The mapping of the different dimensions of the latent representation can be scrambled and the dimensions themselves inverted, producing numerous variants of the original input.

4.4 Hardware Implementation

Currently, we are developing a synthetic body for the halldorophone [9] — an electroacoustic feedback string instrument — in the form of a Eurorack–module–based implementation of the convolution algorithm. This generative virtual resonant body is produced from a number of real-world and synthetic responses, creating new hybrid responses where the resonant response of the instrument shifts dynamically in time, affecting the feedback properties of the instrument.

The initial prototype runs in real–time on a Raspberry Zero 2, making the hardware relatively inexpensive and its power consumption minimal. The hardware can also be used as a standalone generative synthesizer.

4.5 Other Applications

We can calculate the similarities between the spectral representations of any two sounds that have been trained together in a corpus, by calculating the Euclidean distance between their latent representations. These similarity judgements can be used for concatenative synthesis, granular synthesis based on similarities between spectral frames (such as in Bitton et al. [10]), as well as for larger scale compositional judgements based on similarities between sounds.

5. ETHICS

This research presents an interesting ethical question: what does it mean when you borrow and manipulate the aura of a sound or a passage of music and reconstruct it it separately from its source? As none of the original source is present in the output it becomes less of a legal issue although it is still unclear how manipulating borrowed material in this way should be treated under copyright law. We acknowledge the new ethical and legal issues that machine learning in the arts present to us and embrace any future collaboration between artists, engineers, copyright lawyers, and ethicists, discussing the core issues of how new intelligent technologies affect artistic expression.

6. CONCLUSIONS

The Autocoder has shown itself to be a useful tool in recent work by us and our collaborators. Real-world use cases have so far included extending few second long instrumental sounds into multi minute drones that retain the dynamicity and variability of the original instruments; as convolution carrier generators — in one instance allowing us to run inference on twenty models concurrently to construct large dynamic multi-point resonant spaces in real-time; in another case as a tool to morph instrument, voice and natural sound via cross synthesis in an installation setting; and as dynamic filters in the paths of feedback instruments, making them both self-generating musical objects, as well as constantly changing performance instruments. We are pleased with the results, the generated sound often proving surprisingly rich and sophisticated, supporting our belief that there is much potential in our approach.

Listener response to the generated materials makes us believe that source specific modeling represents an important niche in musical machine learning that presents artists with an important tool to expand their DSP toolkits.

We encourage people to download the software and apply the Autocoder in their own work. It is open source under a very permissable license and available for anyone to use and extend. Our future work will focus on the expressive potential and use cases of the Autocoder, further developing the core algorithm and reflect upon its creative use, particularly its potential to be a ghost in the machine within otherwise traditional instruments.

7. REFERENCES

[1] D. B. Franzson, "Autocoder," http://github.com/franzson, 2021.

[2] M. A. Kramer, "Nonlinear Principal Component Analysis using Autoassociative Neural Networks," *AIChE Journal*, no. 2, pp. 233–243, Feb.

[3] D. P. Kingma and M. Welling, "An Introduction to Variational Autoencoders," *Foundations and Trends® in Machine Learning*, no. 4, pp. 307–392.

[4] A. Sarroff and M. Casey, "Musical Audio Synthesis using Autoencoding Neural Nets," *Proceedings ICMC*, 01 2014.

[5] J. Colonel, C. Curro, and S. Keene, "Autoencoding Neural Networks as Musical Audio Synthesizers," *ArXiv*, vol. abs/2004.13172, 2020.

[6] P. Esling, A. Chemla-Romeu-Santos, and A. Bitton, "Generative Timbre Spaces: Regularizing Variational Auto-Encoders with Perceptual Metrics," *CoRR*, 2018.

[7] A. Caillon and P. Esling, "RAVE: A variational autoencoder for fast and high-quality neural audio synthesis," 2021. [Online]. Available: https://arxiv.org/abs/2111.05011

[8] J. Engel, L. H. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable Digital Signal Processing," in *International Conference on Learning Representations*, 2020.

[9] H. Úlfarsson, in *Proceedings of the International Conference on New Interfaces for Musical Expression*, Blacksburg, Virginia, USA.

[10] A. Bitton, P. Esling, and T. Harada, "Neural Granular Sound Synthesis," *CoRR*.