Cloning Guitar Effects with a Recurrent Neural Network

Christian Hassler

1 Abstract

This paper details the implementation of a simple recurrent neural network model which learns to mimic the sonic characteristics of nonlinear audio processing effects. We find that while "distortion" and "fuzz" effects can be efficiently reproduced with extremely minimal resource requirements, other effects such as phase modulation and delay require a more advanced model.

2 Introduction

Creating digital reproductions or "clones" of analog signal processing equipment tends to be process which requires a high degree of specialized knowledge and skill. Beyond analyzing circuits in order to translate their mathematical structure to the digital domain, in order to produce a convincing replica it is often necessary to model the electrical dynamics and idiosyncrasies of individual components. In recent years, machine learning has provided a compelling alternative to this difficult and labor intensive process. By treating a piece of analog equipment as an opaque function of its inputs and outputs, we can train a neural network to find a mapping which reproduces its sonic characteristics automatically.

3 Related Work

Much of this paper is a reproduction of the work done by Wright et al.[8][7] in the area of guitar amplifier modeling. Mehri et al. provided a foundational text for applying recurrent networks to sample-level audio generation with their SampleRNN[4]. In the areas of audio generation and signal processing, recurrent models are frequently rivaled by models based on WaveNet-style dilated temporal convolutions[5][1].

4 Methodology

4.1 Data

Training data was obtained by processing audio clips using a variety of LV2 effects plugins. To assess the model's ability to learn simple nonlinear effects, we used the "MB Distortion" and "Muff"[1] plugins from the free and open source Guitarix[2] plugin collection. The "Phaser" plugin from Guitarix, and the "Dumb Delay" plugin from the LV2 reference plugin suite were also used, in order to investigate the model's ability to learn effects with relatively long time dependencies. In principle, the processing could also be done using analog hardware (and indeed this would be a much more useful application of our model) but digital signal processing was chosen in order to simplify the collection of training data. Unprocessed "source" audio clips were taken from the NSynth[3] dataset, which consists of an annotated collection of 305,979 musical notes, played on a variety of acoustic and electronic instruments.

4.2 Methods

Our model consists of a single LSTM layer followed by a fully connected layer with linear activation. At each time step, the fully connected layer maps the hidden state of the LSTM to a single output sample. This is then summed with the input signal. We used the same loss function as Wright et al.[8]

$$\varepsilon = \varepsilon_{\text{ESR}} + \varepsilon_{\text{DC}}.$$

The first component of this loss function is the error-signal ratio, given by

$$\varepsilon_{\text{ESR}} = \frac{\sum_{i=1}^{n} |y_i - t_i|^2}{\sum_{i=1}^{n} t_i^2}$$

where n is the batch size, and t_i and y_i are the target and predicted output samples, respectively, at index i. Contrary to mean square error, this loss function compensates for the energy of a given signal, so that training is not dominated by high-energy examples. The ESR loss is summed with a second component, which represents the "DC bias" between the prediction and target samples

$$\varepsilon_{\text{DC}} = \frac{\left|\frac{1}{n}\sum_{i=1}^{n}(y_i - t_i)\right|^2}{\frac{1}{n}\sum_{i=1}^{n} t_i^2}.$$

5 Experiments

We randomly selected 10,000 audio clips from the NSynth dataset, and split them into training and validation sets (90% and 10%, respectively). The network was trained for 50 epochs, and the LSTM layer had 16 hidden channels

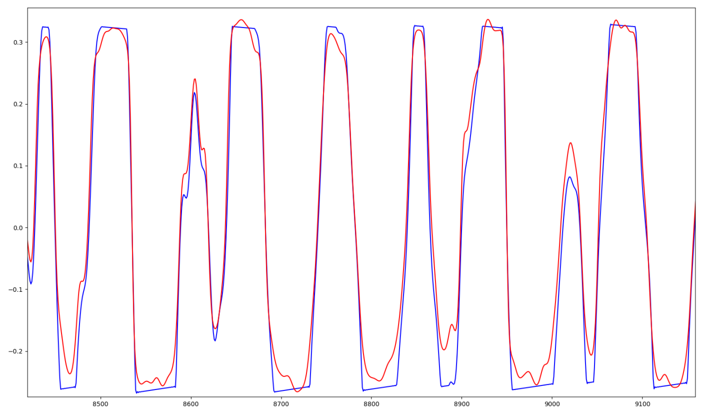
[1]Based on the Electro-Harmonix *Big Muff Pi*, a popular guitar fuzz pedal.

and a depth of 1. Some preliminary experiments were made with slightly larger networks (2 layers with 32 hidden channels) but this did not result in an appreciable improvement in accuracy. All training was conducted on a single Nvidia RTX 2070 graphics card, with 8GB of VRAM.

5.1 Learning nonlinear distortion effects

Our best results were obtained from training on audio samples processed using "distortion" effects. The final validation loss was $\varepsilon = 0.1250$, which corresponds to a mean square error loss of 0.0003. Figures 1 and 2 show sections of audio taken from a model trained on the "Muff" fuzz pedal plugin for 50 epochs. The network consistently fails to accurately reproduce the transient, but converges to the target sequence after a few thousand samples.

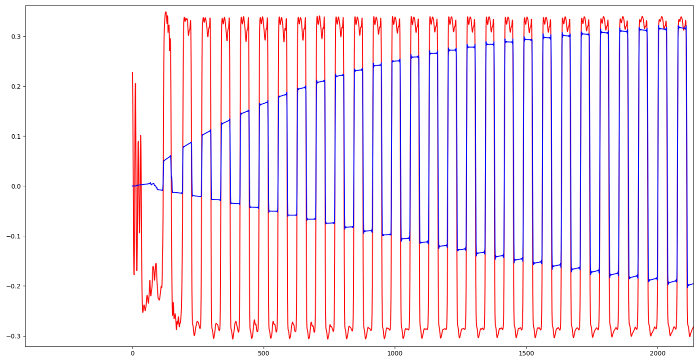
Figure 1: Comparison between target audio (blue) and predicted audio generated by a network which was trained to model a distortion effect (red), taken from the middle of a typical 64000 sample clip.



5.2 Learning modulation and other time-based effects

The model was much less successful at learning effects with relatively long time dependencies. We attempted to train the network to model a 10 Hz "phaser" effect (see Figure 3), and a simple 20 ms delay effect with feedback (see Figure 4). While the model did manage to reproduce some of the finer characteristics of the target "phaser" audio, it has apparently not learned to reproduce the phase modulation itself. In the case of the delay, the model failed to learn anything

Figure 2: Comparison between target audio (blue) and predicted audio generated by a network which was trained to model a distortion effect (red). The network is unsuccessful in modeling the transient.



useful from the training data. Final validation losses for the phaser and delay models were 0.522 and 1.055, respectively.

6 Discussion

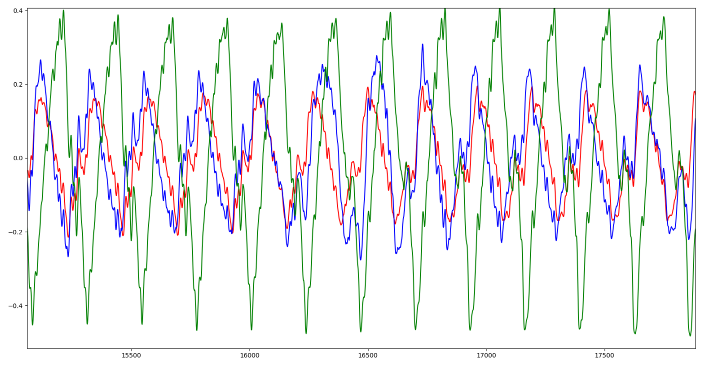
The clear weakness of this architecture is its inability to learn the “slow” temporal relationships, that characterize modulation and delay effects. One possible solution would be to use a “tiered” architecture[4] with each tier operating at a different temporal resolution. Another interesting option would be make use of an intermediate representation in an encoder-decoder architecture. The WaveNet autoencoder developed by Engel et al.[3] takes this approach, and is able to model low frequency features in the training data despite its relatively simple structure. Finally, there has recently been much promising work related to attention mechanisms, and their ability to disentangle the complicated structure of spoken language[6]. A similar approach may be productive here.

References

[1] Jan Chorowski, Ron J. Weiss, Samy Bengio, and Aäron van den Oord. Unsupervised speech representation learning using wavenet autoencoders. *CoRR*, abs/1901.08810, 2019.

[2] Andreas Degert. Guitarix, 2021.

Figure 3: Comparison between target audio (blue), source audio (green), and predicted audio generated by a network which was trained to model a phaser effect (red). Note the drift in phase between the red and blue signals.



[3] Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Douglas Eck, Karen Simonyan, and Mohammad Norouzi. Neural audio synthesis of musical notes with wavenet autoencoders, 2017.

[4] Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron Courville, and Yoshua Bengio. Samplernn: An unconditional end-to-end neural audio generation model, 2017.

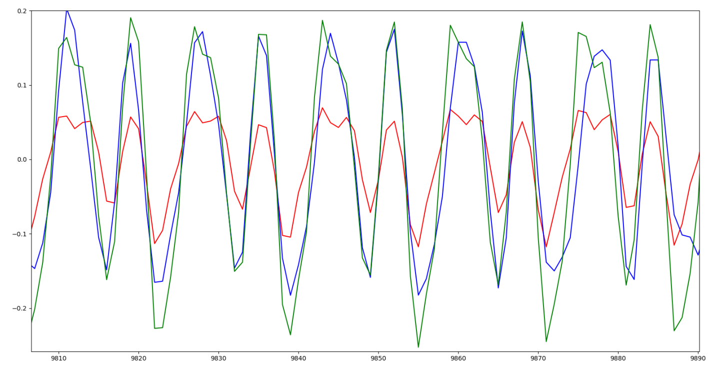
[5] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio, 2016.

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[7] Alec Wright, Eero-Pekka Damskägg, and Vesa Välimäki. Real-time black-box modelling with recurrent neural networks. In *Proceedings of the International Conference on Digital Audio Effects*, Proceedings of the International Conference on Digital Audio Effects. University of Birmingham, September 2019. International Conference on Digital Audio Effects, DAFX ; Conference date: 02-09-2019 Through 06-09-2019.

[8] Alec Wright, Eero-Pekka Damskägg, Lauri Juvela, and Vesa Välimäki. Real-time guitar amplifier emulation with deep learning. *Applied Sciences*, 10(3), 2020.

Figure 4: Comparison between target audio (blue), source audio (green), and predicted audio generated by a network which was trained to model a delay effect (red). The target and source appear similar because the delay effect is not visible on this time scale.



0.2
0.1
0.0
−0.1
−0.2
9810
9820
9830
9840
9850
9860
9870
9880
9890