
Randomized Overdrive Neural Networks

Christian J. Steinmetz
Queen Mary University of London
c.j.steinmetz@qmul.ac.uk

Joshua D. Reiss
Queen Mary University of London
joshua.reiss@qmul.ac.uk

Abstract

By processing audio signals in the time-domain with randomly weighted temporal convolutional networks (TCNs), we uncover a wide range of novel, yet controllable overdrive effects. We discover that architectural aspects, such as the depth of the network, the kernel size, the number of channels, the activation function, as well as the weight initialization, all have a clear impact on the sonic character of the resultant effect, without the need for training. In practice, these effects range from conventional overdrive and distortion, to more extreme effects, as the receptive field grows, similar to a fusion of distortion, equalization, delay, and reverb. To enable use by musicians and producers, we provide a real-time plugin implementation. This allows users to dynamically design networks, listening to the results in real-time. We provide a demonstration and code at <https://ronn.ml>.

1 Introduction

Throughout the history of audio technology, engineers, circuit designers, and particularly guitarists, have searched for novel sonic effects as a result of clipping or distorting audio signals. These distortion effects were first discovered by pushing early guitar amplifiers beyond their operating range, or, in some cases, from the accidental damage to amplifiers or speakers [13]. These pursuits are a clear example of creators taking advantage of the limitations of their tools for creative effect. Distortion effects have permeated many genres such as blues, jazz, rock, and metal, and also play a central role in modern pop and hip-hop styles. Whether it be vacuum tubes, diodes, integrated circuits, or software-based digital models, it appears as if nearly all the methods of generating distortion-based effects have been exhausted. We claim this may not be the case, as we will examine the potential of neural networks for audio signal processing to generate a new class of distortion-based effects.

Neural networks are far from new. In fact, they arose in the same era that blues guitarists began their experiments with distortion [11]. Yet, only more recently, following the emergence of modern deep learning approaches, have neural networks become feasible for audio signal processing [10]. Interestingly, these methods have shown to be successful in emulating the characteristics of amplifiers and distortion effects [12, 14, 2, 6]. While these approaches have been successful in the emulation task, our aim deviates from these virtual analog effect modeling approaches. In a similar spirit to the guitarists who used their amplifiers in a fashion unintended by the original designer, we propose the apparent abuse of neural networks. By utilizing randomly initialized networks as complex signal processing devices, we aim to distort, transform, and warp audio signals for creative effect.

2 Method

2.1 Architecture

We select a convolutional architecture as it provides a parameter efficient method for processing arbitrary length sequences. From the perspective of audio signal processing, it can be considered a series of filters and nonlinear waveshapers. The temporal convolutional network (TCN) formalizes the application of convolutional models operating on 1-dimensional sequences, and outlines a set

of design choices ideal for various sequence modeling tasks [1]. Causal convolutions are a core component of this formulation, in which outputs are predicted considering only past values, so information from the future does not "leak" into current predictions. Additionally, these networks are generally built to be fully convolutional, so they can process input signals of arbitrary length, and will produce an output signal with length proportional to the input length [5].

With standard causal convolutions the receptive field grows linearly as the depth of the network increases. This makes it challenging to achieve models that are able to consider larger time contexts. To address this, the TCN incorporates dilated convolutions, which inserts zeros within the taps of the convolutional kernels, effectively increasing the size of the kernel without additional computation [8]. To increase the receptive field more rapidly, an exponentially increasing dilation factor is generally applied at each layer in the network. We omit residual connections since they are generally used at each layer to stabilize gradient flow, and we do not aim to train these networks [3]. This simplifies the overall architecture, which can then be viewed as a series connection of blocks containing 1-dimensional convolutions followed by a nonlinear activation, as shown in Figure 2.

2.2 Implementation

For the real-time implementation, `ronn`, we utilize the JUCE framework¹, which enables us to create a VST/AU plugin for use in popular digital audio workstations (DAWs). In order to construct the TCN models, we utilize PyTorch [9], which features a C++ API. This enabled us to develop our own parameterized neural network module class that can be instantiated within the main JUCE plugin. By connecting this class with the user interface, the on-screen controls, as shown in Figure 1, can be used to dynamically construct new networks, all in a paradigm that allows for real-time interaction.

A challenge arises since these models are fully convolutional, yet the plugin requires that all processing occur on a block-by-block basis. To address this, we construct a look-back buffer large enough so that the output sequence matches the length of the input block, as demonstrated in Figure 2. In practice, we found this approach produces no perceivable discontinuities at the frame boundaries. As expected, as the size of the receptive field increases, the computational load increases, causing the plugin to perform in less than real-time. We introduce the ability to swap traditional convolutions with depthwise convolutions [4], which reduce the computational overhead by convolving K filters with a single channel each. Using this approach, we were able to run models with larger receptive field, up to 4 seconds, in real-time on the CPU, making a wider range of effects achievable on general purpose hardware utilized by musicians and producers.

Using only a few layers, we achieve subtle to heavy distortion effects similar to traditional approaches, but by stacking more layers and expanding the kernel size, delay-like effects emerge. Using even larger receptive fields produces extreme temporal smearing, similar to reverberation. Adjusting the shape of the nonlinearities also has a significant effect on the timbre of the distortion produced. We found that sigmoid activations often produced very harsh and gritty results, while ReLU activations [7] produced fuzz-like effects. Depending on the depth of the network, we found the weight initialization scheme also impacts the timbre of distortion. Since the TCN can produce any number of output channels, we can generate a stereo output signal, given only a mono input signal. This also enables cross-channel interactions for stereo inputs. We found that these configurations often produced interesting spatialized results. Finally, a global seed control enables effect recallability and presets.

3 Discussion

We propose the use of randomly weighted TCNs as complex, time-domain audio signal processing devices. We find that adjusting various architectural aspects results in the ability to generate a wide range of compelling sonic effects. This presents a new paradigm for designing audio effects. Instead of adjusting the controls of traditional processors, users can explore a wider space of effects by adjusting the architecture of a neural network. While deep networks pose a challenge for real-time implementation due to significant compute overhead, we overcome this with some careful design choices. With a simplified interface, musicians and producers without machine learning experience can easily take advantage of these effects. While we have only investigated feedforward architectures, it reasons that we could achieve a wider range of effects with the addition of recurrent pathways.

¹<https://github.com/juce-framework/JUCE>

Broader Impact

In this work, we applied existing neural network approaches for processing audio signals in a creative context. Since we did not utilize any training data in this process, there is a relatively low risk of bias arising here. Potential biases in the processing of signals may arise from the randomized network weights, which are sampled from various common distributions. Any current biases reflect the underlying characteristics of these generated effects. Future work could investigate potential biases in different weight initialization schemes used when processing different sources.

References

- [1] Shaojie Bai, J. Sizo Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv:1803.01271*, 2018.
- [2] Eero-Pekka Damskägg, Lauri Juvela, Vesa Välimäki, et al. Real-time modeling of audio distortion circuits with deep learning. In *Sound and Music Computing Conf. (SMC)*, pages 332–339, 2019.
- [3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.
- [4] Andrew G. Howard, Menglong Zhu, Bo Chen, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv:1704.04861*, 2017.
- [5] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.
- [6] Marco A. Martínez Ramírez and Joshua D. Reiss. Modeling nonlinear audio effects with end-to-end deep neural networks. In *IEEE Int Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 171–175, 2019.
- [7] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, pages 807–814, 2010.
- [8] Aaron van den Oord, Sander Dieleman, Heiga Zen, et al. Wavenet: A generative model for raw audio. *arXiv:1609.03499*, 2016.
- [9] Adam Paszke, Sam Gross, Francisco Massa, et al. Pytorch: an imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, pages 8024–8035. Curran Associates, Inc., 2019.
- [10] Hendrik Purwins, Bo Li, Tuomas Virtanen, Jan Schlüter, Shuo-Yiin Chang, and Tara Sainath. Deep learning for audio signal processing. *IEEE Journal of Selected Topics in Signal Processing*, 13(2):206–219, 2019.
- [11] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.
- [12] Thomas Schmitz and Jean-Jacques Embrechts. Nonlinear real-time emulation of a tube amplifier with a long short time memory neural-network. In *144th Convention of the Audio Engineering Society (AES)*, 2018.
- [13] John Shepherd, David Horn, Dave Laing, Paul Oliver, and Peter Wicke, editors. *Continuum Encyclopedia of Popular Music of the World: Performance and production. Volume 2*, page 286. Encyclopedia of Popular Music of the World. Bloomsbury Academic, 2003.
- [14] Zhichen Zhang, Edward Olbrych, Joseph Bruchalski, Thomas J. McCormick, and David L. Livingston. A vacuum-tube guitar amplifier model using long/short-term memory networks. In *Proc. of IEEE Southeastcon*, pages 1–5, 2018.

Supplementary materials

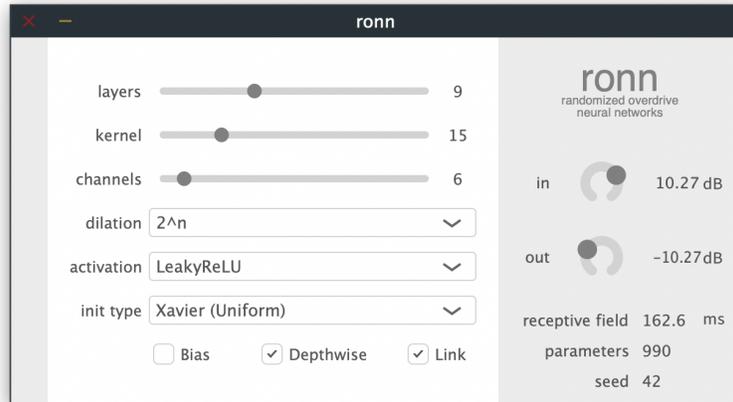


Figure 1: The real-time plugin user interface featuring a series of sliders and selection boxes, enabling users to dynamically construct various temporal convolutional network (TCN) architectures while listening to the results. When the user adjusts any of the on-screen controls, a callback will run, constructing a neural network with the new architectural design. On the bottom right, indicators show the receptive field in milliseconds, the number of parameters in the network, and the global seed.

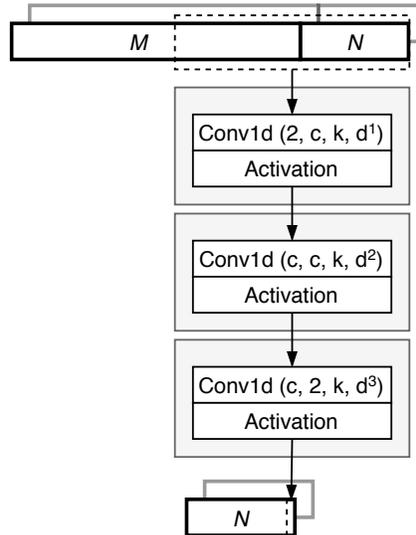


Figure 2: Block diagram outlining the block-based processing as well as a 3-layer network in a stereo input-output configuration. Each block within the network consists only of a 1-dimensional convolution followed by an activation function. The look-back buffer is shown at the top, which consists of N samples from the current input block, concatenated with M past input samples. The number of stored past samples M is a function of the receptive field of the TCN, shown in the dotted box, and the block size N . M is selected such that the entire buffer of size $M + N$ will produce an output of N samples. Recall that since padding is not used, the output of each convolutional block will be smaller than the input. When the receptive field is quite large, producing the output block (e.g. ≈ 10 ms) may require processing multiple seconds of audio.