Audio Engineering Society
Convention Express Paper 33

Presented at the 153rd Convention
2022 October

Approximating Ballistics in a Differentiable Dynamic Range Compressor

Joseph T. Colonel[1] and Joshua Reiss[1]

[1]*Queen Mary University of London*

Correspondence should be addressed to Joseph T. Colonel (j.t.colonel@qmul.ac.uk)

ABSTRACT

We present a dynamic range compressor with ballistics implemented in a differentiable framework that can be used for differentiable digital signal processing tasks. This compressor can update the values of its threshold, compression ratio, knee width, makeup gain, attack time, and release time using stochastic gradient descent and backpropagation techniques. The performance of this technique is evaluated on a reverse engineering of audio effects task, in which the parameter settings of a dynamic range compressor are inferred from a dry and wet pair of audio samples. Techniques for initializing the parameter estimates in this reverse engineering task are outlined and discussed.

1 Introduction

Dynamic range compression is an essential tool in mix engineering and audio production [1]. This nonlinear audio effect is often used to reduce the dynamic range of a signal. This is done by selectively attenuating peaks in the signal while leaving quieter portions untouched. Digital implementations of dynamic range compressors (DRC) vary, leading to much diversity in the types of controls exposed to users and the considerations taken into account when setting parameter values [2].

Challenges arise in the design of digital DRC as it is a nonlinear time-dependent audio effect with memory. A typical feedforward digital DRC, as outlined in [3], is composed of the following modules: a level detector tasked with measuring the level of the incoming signal; a gain computer tasked with calculating the instantaneous attenuation or gain to be applied to the signal; and a gain smoother tasked with modifying the output of the gain computer based on the time-varying fluctuations of the input signal's loudness. Key to the design of many DRCs are the attack and release times that control the gain smoother. These parameters help to avoid introducing distortion and control how quickly the DRC acts.

Due to the complexity of a DRC and its usage, much work has gone into characterizing, estimating, and automating its parameters. Previous literature has developed procedures and test signals for profiling DRCs [3], used regression models and reference signals to control a DRC [4], developed feature-based heuristics

to operate a DRC [5], and implemented cross-adaptive algorithms to set DRC parameters settings across a multitrack recording [6].

Parallel to this work is the field of differentiable digital signal processing (DDSP) [7], in which common DSP modules are manually implemented in a differentiable framework such as Tensorflow or Pytorch [8, 9]. This autodifferentiated regime allows for these modules to be implemented in or controlled by neural networks due to their ability to backpropagate gradients. Individual effects such as parametric EQs [10, 11], reverberation [12], and distortions have been implemented [13]. DDSP has found usage in tasks such as audio synthesis [14, 15, 16], singing voice synthesis [17, 18], and reverse engineering audio effects [19].

This work is organized as follows. Section 2 details common parameters to control a DRC and previous approaches to implementing a differentiable DRC. Section 3 outlines the implementation of a DDSP DRC based on the design presented in [3]. To our knowledge this is the first such implementation with uncoupled attack and release times. Section 4 presents an evaluation of the proposed method on test signals and on a reverse engineering of audio effects task. Finally, section 5 discusses the performance of the DRC.

2 Background

2.1 Dynamic Range Compressor Parameters

The operation of a DRC can be defined using the following parameters. The analysis presented here is primarily adapted from [2].

Threshold is the level used to determine whether or not to apply compression to the input signal. When the signal is measured above the threshold, compression is applied. In this formulation, the knee is not centered about the threshold; instead, it begins after the threshold. How the input's level is measured varies across DRC designs, with typical implementations including sample-by-sample peak detection and root-mean-square (RMS) measures.

Ratio determines the amount of compression applied and is a measure of the input/output ratio for signals crossing the threshold.

Knee-width is a threshold-dependent value that allows for a smooth transition in the DRC's compression characteristic curve above and below the threshold. A small value creates a sharp transition between unity gain and the compression ratio, and a larger value produces a gradual transition.

Makeup Gain refers to a gain applied to the compressor's output signal.

Attack time and *release time* determine how long it takes the compressor to attenuate the signal according to the ratio after surpassing the threshold and how long the compressor continues to attenuate the signal after dropping below the threshold, respectively. In many designs these parameters also control how the attenuation applied by the compressor is smoothed over time.

2.2 Differentiable Compressors

While blackbox neural networks have been used to emulate DRCs [20, 21], to the authors' knowledge only one DDSP DRC has been proposed in the literature [22]. There the authors implemented a DRC using Pytorch with tunable threshold, ratio, knee width, makeup gain, and a ballistics control. The authors approximate both attack and release time with a joint smoothing parameter that controls a single pole IIR filter to smooth the DRC's attenuation curve. This IIR filter is then approximated using an FIR filter. As stated by the authors, forcing the attack time and release time to be shared restricts the modelling capabilities of the DRC.

As stated in [22], the difficulty in implementing a DRC using a differentiable framework lies in the recursive nature of the attack and release calculation. This sample-by-sample "differentiation through time" is costly in both time and memory. Thus the methodology presented here seeks to avoid sample-by-sample calculations and instead approximates attack and release passages as smoothing filters applied to a downsampled loudness curve of the audio signal.

3 Proposed Method

Refer to Figure 1 for a block diagram of the proposed system. Given a fixed length audio signal sampled at 44.1kHz and values for parameters mentioned in Section 2.1 the following steps are used to apply dynamic range compression.

First, a root mean square (RMS) level measurement is calculated using a 5ms window and hop size 0.22ms and converted to dB. This generates a loudness curve measured at 4410 frames per second.

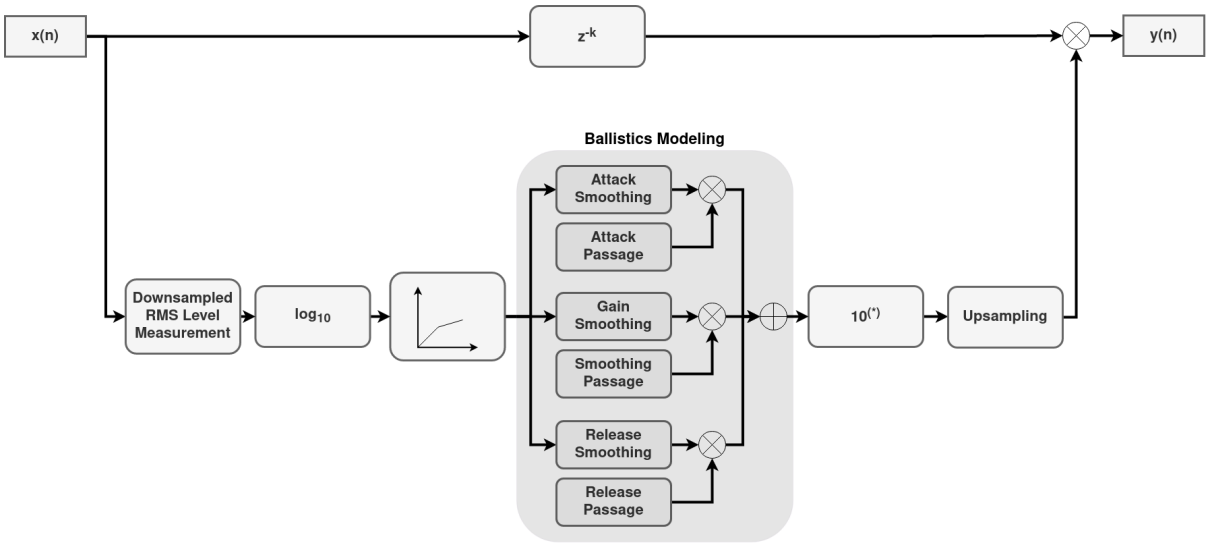

Fig. 1: Block diagram of proposed system

Then, attack and release passages are estimated by finding when this loudness curve crosses the threshold value. Attack passages are calculated by convolving a rectangular window of length τ_{at} with the rising edge of the input signal passing the threshold, and release passages by convolving a rectangular window of length τ_{rt} with the falling edge. The length of these rectangular windows correspond to the attack and release times calculated in frames. These passages finally interfere with one another when they overlap so that the DRC is not simultaneously set to attack and release. Finally, gain smoothing passages are calculated by finding the portions of the loudness curve both above the threshold and outside of attack passages. Thus three masks are produced that are the length of the signal's loudness curve corresponding to attack passages, release passages, and smoothing passages.

Afterwards a compression characteristic is calculated using the threshold, ratio, and knee width for the duration of the signal. This compression characteristic curve is then subtracted from the original signal's loudness curve in order to produce an attenuation curve. Note that this curve measures the dB attenuation per frame that when applied to the original signal produces the characteristic curve.

Given a time constant τ in frames, an approximate moving average filter with support $[0,N]$ takes the form

$$h(x) = \frac{1}{\sum_0^N (\tanh(0.1 * \mathrm{relu}(\tau - x))} \tanh(0.1 * \mathrm{relu}(\tau - x)) \tag{1}$$

where $\tanh(x)$ refers to the hyperbolic tangent function and $\mathrm{relu}(x)$ refers to the rectified linear unit. While a multiplicative constant larger than 0.1 would make h(x) more closely approximate a moving average filter, it was experimentally found that the 0.1 scaling factor provides a decent approximation while allowing for gradients to backpropagate through the system.

Three approximate moving average filters are calculated using τ_{at}, τ_{rt}, and τ_{st}, corresponding to the attack action, release action, and gain smoothing action of the DRC. These three filters are convolved in parallel with the attenuation curve, windowed according to the attack/release/smoothing passages mentioned above, and then summed. Afterwards the makeup gain is applied.

Finally the smoothed attenuation curve is converted from dB to a linear scale, upsampled from 4410 frames per second to the original sampling rate using linear interpolation, delayed by 5ms to simulate the lag in level measurement, and applied to the original audio sample via multiplication.

Table 1: DRC parameters used in ReaComp VST and learned from gradient descent on test signal.

	ReaComp	Learned Value
Threshold (dB)	-10.0	-11.7
Ratio	10.0	4.8
Makeup Gain (dB)	-3.0	-3.0
Knee Width (dB)	0.0	1.6
Attack Time (ms)	40.0	54.0
Release Time (ms)	200.0	268.0
Smoothing Time (ms)	-	31.1

Table 2: DRC parameters used in ReaComp VST and learned from gradient descent on speech signal.

	ReaComp	Learned Value
Threshold (dB)	-15.0	-20.5
Ratio	5.0	1.8
Makeup Gain (dB)	3.9	3.4
Knee Width (dB)	1.0	0.0
Attack Time (ms)	3.0	35.1
Release Time (ms)	300.0	37.6
Smoothing Time (ms)	-	13.2

4 Reverse Engineering Dynamic Range Compression

The modeling capability of the proposed method is evaluated using a reverse engineering of dynamic range compression task [3], in which the parameters of a DRC are inferred using a compressed signal and its dry counterpart. A similar task was proposed in [23], where dynamic effects processing in a multitrack mix was estimated using frame-based polynomial gain estimation. Two signals were chosen to test the performance of the method: a signal proposed in [3] to profile the ballistics of a DRC, and a clip of speech. Compressed signals were generated using the Cockos VST ReaComp plugin with an RMS level detector set to 5ms.

4.1 Initialization and Optimization

Because the methodology proposed in Section 3 can be implemented in an autodifferentiating framework such as Tensorflow, a gradient descent can be performed to optimize DRC parameters for a given dry/wet audio pair. The cost is calculated by passing a dry audio sample through the estimated DRC and measuring the distance between the estimated and target wet audio signal. Gradient descent is performed using the Adam method with an initial learning rate of 10^{-4} [24]. The cost function chosen is multiscale spectrogram loss with window sizes 46ms, 12ms, and 3ms [7]. This loss function is chosen to avoid phase issues that may arise. Optimization is allowed to run for a maximum of 40000 iterations. Early stopping is employed with a patience of 1000 iterations.

The DRC parameters must be initialized such that each contributes to the compression applied to the dry signal. Otherwise, these parameters will not update during optimization. Furthermore, "reasonable" parameters should be chosen to avoid portions of the loss surface very far from expected DRC parameters. As such the threshold value is initialized close to the mean value of the dry signal's downsampled RMS level curve, the ratio initialized close to 2.0, the knee-width initialized close to 2dB, makeup gain initialized just above 0dB, τ_{at} and τ_{st} initialized close to 45 frames (about 10ms), and τ_{rt} initialized close to 450 frames (about 100ms).

4.2 Results

Tables 1 and 2 compare the VST plugin settings to the parameters learned in the gradient descent. Figures 2 and 4 show the uncompressed, VST compressed, and differentiable DRC compressed waveforms for the test signal and speech signal respectively. Figures 3 and 5 show the downsampled loudness curves of the VST compressed and differentiable DRC compressed waveforms for the test signal and speech signal respectively.

5 Discussion

In general it is difficult to compare parameter settings across DRCs as their implementations vary greatly across designs. Furthermore, few DRCs have an explicit τ_{st} to measure. As such attention will be paid to the compressed waveforms themselves.

With the test signal, the differentiable DRC is able to match the static characteristic of the target signal well, with nearly 0dB residual. Though the learned release time is off by 68ms from the VST, the differentiable DRC's release passage closely matches that of

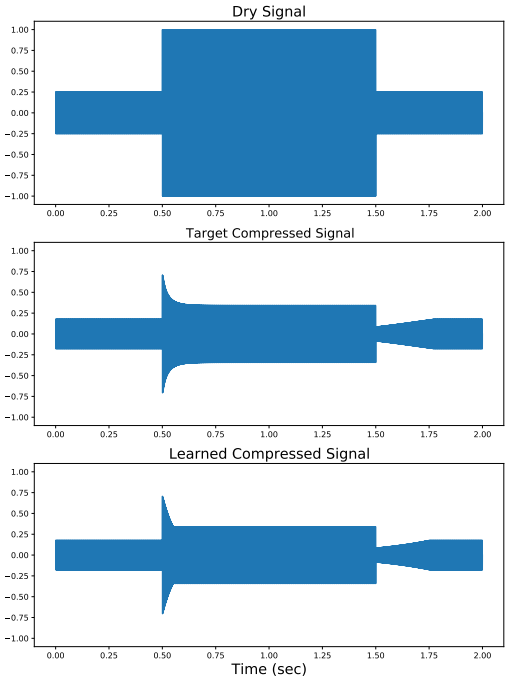

Fig. 2: Test signal waveforms

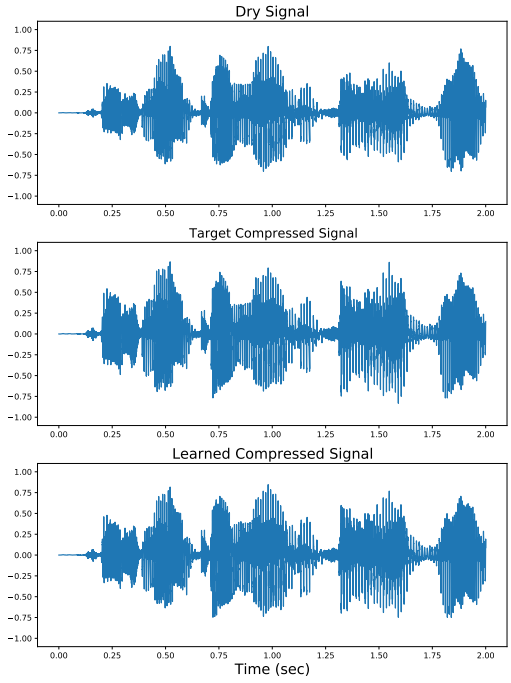

Fig. 4: Speech signal waveforms

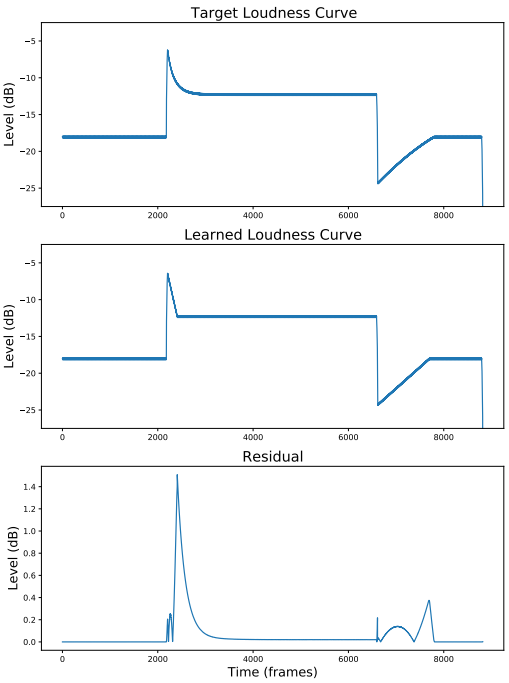

Fig. 3: Test signal loudness curves

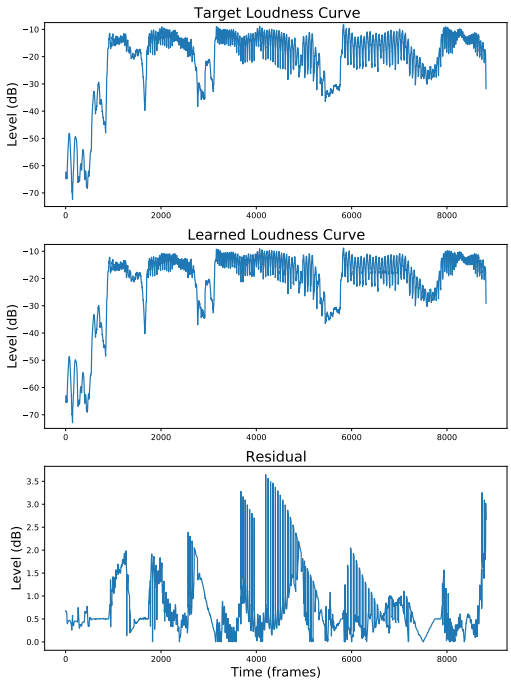

Fig. 5: Speech signal loudness curves

the VST's. The biggest discrepancy occurs during the attack passage, where the differentiable DRC cannot match the VST's sloped attack attenuation settling.

The speech signal tasks the differentiable DRC with matching a "light touch" compression on a dynamic signal, which it is able to do within about 3.5dB. Though the differentiable DRC's threshold is set several dB lower than the VST, it compensates with a longer attack time — this smoothing decreases the initial attenuation to portions of the loudness curve just above -20dB. It is interesting to note that the differentiable DRC learns a makeup gain 0.5dB smaller than the target. Better initialization may help avoid this bias and improve the parameter estimation.

6 Conclusion

We have presented a method for implementing a dynamic range compressor with attack and release in an autodifferentiating framework. This is achieved by convolving approximate moving average filters with a characteristic attenuation curve, windowing, and then summing to simulate the attack smoothing, release smoothing, and overall gain smoothing found in typical dynamic range compressor designs. The performance of this differentiable dynamic range compressor is baselined on a reverse engineering of compression task.

7 Acknowledgement

This work is partly supported by the Research and Development Division of Yamaha Corporation, Japan.

References

[1] Izhaki, R., *Mixing audio: concepts, practices, and tools*, Focal Press, 2008.

[2] Giannoulis, D., Massberg, M., and Reiss, J. D., "Digital dynamic range compressor design—A tutorial and analysis," *Journal of the Audio Engineering Society*, 60(6), pp. 399–408, 2012.

[3] Bitzer, J., Schmidt, D., and Simmer, U., "Parameter estimation of dynamic range compressors: models, procedures and test signals," in *Audio Engineering Society Convention 120*, Audio Engineering Society, 2006.

[4] Sheng, D. and Fazekas, G., "Automatic control of the dynamic range compressor using a regression model and a reference sound," in *Proceedings of the 20th International Conference on Digital Audio Effects (DAFx-17)*, pp. 160–167, 2017.

[5] Giannoulis, D., Massberg, M., and Reiss, J. D., "Parameter automation in a dynamic range compressor," *Journal of the Audio Engineering Society*, 61(10), pp. 716–726, 2013.

[6] Ma, Z., De Man, B., Pestana, P. D., Black, D. A., and Reiss, J. D., "Intelligent multitrack dynamic range compression," *Journal of the Audio Engineering Society*, 63(6), pp. 412–426, 2015.

[7] Engel, J., Gu, C., Roberts, A., et al., "DDSP: Differentiable Digital Signal Processing," in *International Conference on Learning Representations*, 2019.

[8] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al., "TensorFlow: a system for Large-Scale machine learning," in *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pp. 265–283, 2016.

[9] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al., "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, 32, 2019.

[10] Nercessian, S., "Neural parametric equalizer matching using differentiable biquads," 2020.

[11] Colonel, J. T., Steinmetz, C. J., Michelen, M., and Reiss, J. D., "Direct design of biquad filter cascades with deep learning by sampling random polynomials," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3104–3108, IEEE, 2022.

[12] Lee, S., Choi, H.-S., and Lee, K., "Differentiable artificial reverberation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2022.

[13] Nercessian, S., Sarroff, A., and Werner, K. J., "Lightweight and interpretable neural modeling

of an audio distortion effect using hyperconditioned differentiable biquads," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 890–894, IEEE, 2021.

[14] Hayes, B., Saitis, C., and Fazekas, G., "Neural waveshaping synthesis," in *International Society for Music Information Retrieval Conference (ISMIR)*, 2021.

[15] Shan, S., Hantrakul, L., Chen, J., Avent, M., and Trevelyan, D., "Differentiable Wavetable Synthesis," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4598–4602, IEEE, 2022.

[16] Caspe, F., McPherson, A., and Sandler, M., "DDX7: Differentiable FM Synthesis of Musical Instrument Sounds," *arXiv preprint arXiv:2208.06169*, 2022.

[17] Nercessian, S., "End-to-End Zero-Shot Voice Conversion Using a DDSP Vocoder," in *2021 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pp. 1–5, 2021, doi:10.1109/WASPAA52581.2021.9632754.

[18] Nercessian, S., "Differentiable WORLD Synthesizer-based Neural Vocoder With Application To End-To-End Audio Style Transfer," *arXiv preprint arXiv:2208.07282*, 2022.

[19] Colonel, J. T. and Reiss, J., "Reverse engineering of a recording mix with differentiable digital signal processing," *The Journal of the Acoustical Society of America*, 150(1), pp. 608–619, 2021.

[20] Hawley, S., Colburn, B., and Mimilakis, S. I., "Profiling audio compressors with deep neural networks," in *Audio Engineering Society Convention 147*, Audio Engineering Society, 2019.

[21] Steinmetz, C. J. and Reiss, J. D., "Efficient neural networks for real-time modeling of analog dynamic range compression," in *Audio Engineering Society Convention 151*, Audio Engineering Society, 2022.

[22] Steinmetz, C. J., Bryan, N. J., and Reiss, J. D., "Style Transfer of Audio Effects with Differentiable Signal Processing," 2022.

[23] Barchiesi, D. and Reiss, J., "Reverse engineering of a mix," *Journal of the Audio Engineering Society*, 58(7/8), pp. 563–576, 2010.

[24] Kingma, D. P. and Ba, J., "Adam: A Method for Stochastic Optimization," in *ICLR (Poster)*, 2015.