ANTIDERIVATIVE ANTIALIASING FOR RECURRENT NEURAL NETWORKS

Otto Mikkonen and Kurt James Werner

Soundtoys, Inc.
Burlington, VT, USA
{otto|kurt}@soundtoys.com

ABSTRACT

Neural networks have become invaluable for general audio processing tasks, such as virtual analog modeling of nonlinear audio equipment. For sequence modeling tasks in particular, recurrent neural networks (RNNs) have gained widespread adoption in recent years. Their general applicability and effectiveness stems partly from their inherent nonlinearity, which makes them prone to aliasing. Recent work has explored mitigating aliasing by oversampling the network—an approach whose effectiveness is directly linked with the incurred computational costs. This work explores an alternative route by extending the antiderivative antialiasing technique to explicit, computable RNNs. Detailed applications to the Gated Recurrent Unit and Long Short-Term Memory cell are shown as case studies. The proposed technique is evaluated on multiple pre-trained guitar amplifier models, assessing its impact on the amount of aliasing and model tonality. The method is shown to reduce the models' tendency to alias considerably across all considered sample rates while only affecting their tonality moderately, without requiring high oversampling factors. The results of this study can be used to improve sound quality in neural audio processing tasks that employ a suitable class of RNNs. Additional materials are provided in the accompanying webpage[1].

1. INTRODUCTION

Recent years have seen wide adoption of neural networks to a range of audio signal processing tasks, including virtual analog (VA) modeling [1, 2, 3, 4]. In VA modeling—where analog hardware is emulated using digital signal processing techniques—many tasks involve sequence modeling, making recurrent neural networks (RNNs) a natural choice. The use of RNNs is motivated by their real-time capability, parameter efficiency, and low-latency operation [3, 2]

Many of the devices considered in VA modeling are nonlinear or otherwise extend the signal bandwidth, making them susceptible to aliasing. Prominent aliasing distortion can manifest as unpleasant roughness, inharmonicity, or beating, ultimately degrading audio quality [5]. Minimizing aliasing is therefore crucial.

The available methods for mitigating aliasing depend on the properties of the bandwidth-extending operator. While oversampling the system is a conceptually simple and widely applicable approach to reducing aliasing [5], its effectiveness is directly tied to the chosen oversampling factor and, consequently, affects the incurred computational costs. When aliasing results due to discontinuities in the waveform or one of this derivatives, bandlimited correction functions can be employed. Examples of such techniques include the bandlimited step (BLEP) [6] and the bandlimited ramp (BLAMP) [7] functions. Antiderivative antialiasing (ADAA), introduced by Parker et al. [8], is applicable to memoryless nonlinearities, and operates by differentiating the (continuous-time) antiderivatives of the underlying nonlinearity in discrete time. Since its introduction, several extensions have emerged, including applications to general nonlinear state-space systems [9] and piecewise polynomials [10].

In neural audio signal processing, the networks' tendency to alias has been recognized [11], but methods for mitigating the problem have seen less exploration. Vanhatalo et al. [12] formally investigate the issue and evaluate known antialiasing methods applicable to recurrent and convolutional network architectures, including data oversampling, architectural modifications, and training adjustments. Somewhat surprisingly, among the considered methods, only model pruning proved viable—though at the cost of reduced model accuracy. Köper et al. [13] propose an approach for applying ADAA to the state trajectory network (STN) architecture [4]. While otherwise successful, the method requires training the model on specially devised synthetic data. Finally, Carson et al. [1] introduce a general technique for oversampling RNNs, hereafter referred to as sample rate (SR) independent RNNs, and explore its potential for antialiasing. While especially integer factor oversampling was shown to preserve model character while significantly reducing aliasing, the approach comes at the cost of increased computational complexity. A follow-up work investigates data resampling as an alternative [14]. The authors also acknowledge very recent preprints on the topic [15, 16].

This work builds on Holters' state-space ADAA extension [9] and SR independent RNNs [1] to extend the ADAA approach to RNNs. To demonstrate it, we will treat the Gated Recurrent Unit (GRU) [17] and the Long Short-Term Memory (LSTM) [18]. The proposed technique is evaluated on multiple publicly available pretrained guitar amplifier models, assessing its influence on aliasing reduction and model tonality—defined here as the spectrum produced by the non-aliased harmonic components—across different inference rates. The method is shown to effectively reduce aliasing artifacts produced by the models, while only moderately affecting their tonality.

The rest of this paper is organized as follows. Sec. 2 provides the necessary background information. Sec. 3 presents our technique and applies it to the GRU and the LSTM as case studies, and describes its combination with SR independent RNNs. In Sec. 4, the experimental procedure is outlined, with the corresponding results presented in Sec. 5. Finally, Sec. 6 offers concluding remarks and proposes directions for future research.

[1] https://otto-soundtoys.github.io/dafx25-adaa-for-rnns/

2. BACKGROUND

This section introduces the necessary background information for the method. The shorthand notation $\alpha^n \equiv \alpha[n]$ is used to denote the nth sample of sequence α.

2.1. Antiderivative Antialiasing (ADAA)

Consider a discrete-time memoryless nonlinear mapping

$$y^n = f(x^n), \tag{1}$$

where (x, y) are the input and output sequences, respectively, with individual elements ($x^n \in \mathbb{R}, y^n \in \mathbb{R}$), and $f : \mathbb{R} \mapsto \mathbb{R}$. An ideal alias-free variant of Eq. (1) results from applying the same nonlinearity in continuous time, or

$$y(t) = f(x(t)), \tag{2}$$

where $(x(t), y(t))$ are the underlying continuous-time equivalents of sequences (x, y). Parker et al. [8] derive an expression for a discrete-time approximation of Eq. (2) by assuming a piecewise-linear $x(t)$ and applying an antialiasing filter in continuous time using a rectangular kernel. The solution takes the form

$$\overset{\star}{y}{}^n = \overset{\star}{f}_1(x^n, x^{n-1}) = \frac{F_1(x^n) - F_1(x^{n-1})}{x^n - x^{n-1}}, \tag{3}$$

where F_1 is the 1st antiderivative of f and $\star$ is used here to denote an antialiased expression.

Bilbao et al. [19] notice that Eq. (3) approximates the derivative $y(x) = \mathrm{d}F_1/\mathrm{d}x$ and extends to the oth order case by

$$y(x) = \frac{\mathrm{d}^o F_o(x)}{\mathrm{d}x^o} = \mathrm{D}^o F_o(x), \quad \mathrm{D} = \frac{1}{\mathrm{d}x/\mathrm{d}t}\frac{\mathrm{d}}{\mathrm{d}t}, \tag{4}$$

where F_o is the oth antiderivative of f, and D the differential operator. The expression can be evaluated in discrete time by approximating D as a combination of unit forward/backwards shifts and first difference operations, written hereafter simply as

$$\overset{\star}{y}{}^n = \overset{\star}{f}_o(x^n, ..., x^{n-o}) \equiv \overset{\star}{f}_o(x^n, ...), \tag{5}$$

with the detailed form and numerical considerations provided in earlier work [19, 5, 20]. The "..."-notation is used to abstract away the past inputs required for the evaluation for brevity. The method adds memory to an otherwise memoryless system and delays the resulting sequence $\overset{\star}{y}$ in a nonlinear and frequency-dependent fashion that depends on the ADAA order—for linear functions (or linearizations), this can be thought of as a group delay.

Holters' [9] derives a state-space extension to the method to include systems of the form

$$\mathbf{h}^n = \mathbf{A}\mathbf{h}^{n-1} + \mathbf{B}\mathbf{x}^n + f_h(\mathbf{p}_h^n) \tag{6}$$

$$\mathbf{y}^n = \mathbf{C}\mathbf{h}^{n-1} + \mathbf{D}\mathbf{x}^n + f_y(\mathbf{p}_y^n) \tag{7}$$

$$\mathbf{p}_h^n = \mathbf{C}_x\mathbf{h}^{n-1} + \mathbf{D}_x\mathbf{x}^n \tag{8}$$

$$\mathbf{p}_y^n = \mathbf{C}_y\mathbf{h}^{n-1} + \mathbf{D}_y\mathbf{x}^n, \tag{9}$$

where $\mathbf{x}^n \in \mathbb{R}^{\mathrm{X}}, \mathbf{y}^n \in \mathbb{R}^{\mathrm{Y}}$, with (X, Y) denoting the number of input resp. output channels, $\mathbf{h}$ is a sequence of states with elements $\mathbf{h}^n \in \mathbb{R}^{\mathrm{H}}$ with H denoting the state size, $(f_h : \mathbb{R}^{\mathrm{H}} \mapsto \mathbb{R}^{\mathrm{H}}, f_y : \mathbb{R}^{\mathrm{Y}} \mapsto \mathbb{R}^{\mathrm{Y}})$ are two nonlinear functions with inputs $(\mathbf{p}_h : \mathbb{R}^{\mathrm{X}} \times \mathbb{R}^{\mathrm{H}} \mapsto \mathbb{R}^{\mathrm{H}}, \mathbf{p}_y : \mathbb{R}^{\mathrm{X}} \times \mathbb{R}^{\mathrm{H}} \mapsto \mathbb{R}^{\mathrm{Y}})$, respectively, and $(\mathbf{A}, ..., \mathbf{D}_y)$ are the parameter matrices of the system.

Table 1: *Considered synchronization filters.*

$\mathrm{H}_1(\alpha^n, ...) =$	$\mathrm{H}_1(\alpha^n, \alpha^{n-1}) = \frac{\alpha^n}{2} + \frac{\alpha^{n-1}}{2}$
$\mathrm{H}_2(\alpha^n, ...) =$	$\mathrm{H}_2(\alpha^n, \alpha^{n-1}, \alpha^{n-2}) = \frac{\alpha^n}{4} + \frac{\alpha^{n-1}}{2} + \frac{\alpha^{n-2}}{4}$

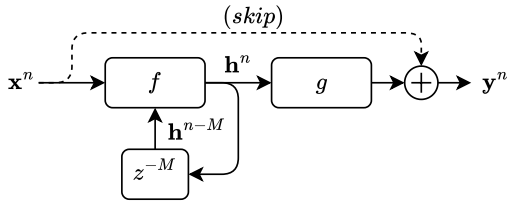


Figure 1: *Considered model architecture.*

The system is antialiased by applying ADAA to the nonlinearities (f_h, f_y) and adding order-dependent synchronization filters $\mathrm{H}_o(\alpha^n, ...)$ for aligning the internal signals, written as

$$\overset{\star}{\mathbf{x}}{}^n = \bar{\mathbf{A}}\mathrm{H}_o(\mathbf{h}^{n-1}, ...) + \bar{\mathbf{B}}\mathrm{H}_o(\mathbf{x}^n, ...) + \overset{\star}{f}_{o,h}(\mathbf{p}_h^n, ...) \tag{10}$$

$$\overset{\star}{\mathbf{y}}{}^n = \bar{\mathbf{C}}\mathrm{H}_o(\mathbf{h}^{n-1}, ...) + \bar{\mathbf{D}}\mathrm{H}_o(\mathbf{x}^n, ...) + \overset{\star}{f}_{o,y}(\mathbf{p}_y^n, ...). \tag{11}$$

Due to the added group delay in the feedback path, the system operates at a higher SR than the original, requiring the use of modified parameter matrices $(\bar{\mathbf{A}}, ..., \bar{\mathbf{D}})$. Two examples of synchronization filters which we use for the $o = 1, 2$ cases are given in Tab. 1. Both filters correspond to the small-signal response of the system, derived via Taylor series expansion [19]. Note well that these filters are written as functions rather than the more typical z-transform notation, for parity with the nonlinear functions involved in ADAA.

2.2. Sample Rate (SR) Independent RNNs

This work deals with network architectures of the form

$$\mathbf{h}^n = f(\mathbf{x}^n, \mathbf{h}^{n-1}) \tag{12}$$

$$\mathbf{y}^n = \begin{cases} g(\mathbf{h}^n) + \mathbf{x}^n & \text{if "skip connection"} \\ g(\mathbf{h}^n) & \text{otherwise,} \end{cases} \tag{13}$$

where $f : \mathbb{R}^{\mathrm{X}} \times \mathbb{R}^{\mathrm{H}} \mapsto \mathbb{R}^{\mathrm{H}}$ is now the nonlinear mapping performed by a recurrent unit, $g : \mathbb{R}^{\mathrm{H}} \mapsto \mathbb{R}^{\mathrm{Y}}$ an affine transformation performed by a fully connected (FC) output layer, H denotes the hidden size, and the rest is as in Sec. 2.1. An optional skip connection links the network input to its output, in which case the network learns the residual function [11, 4].

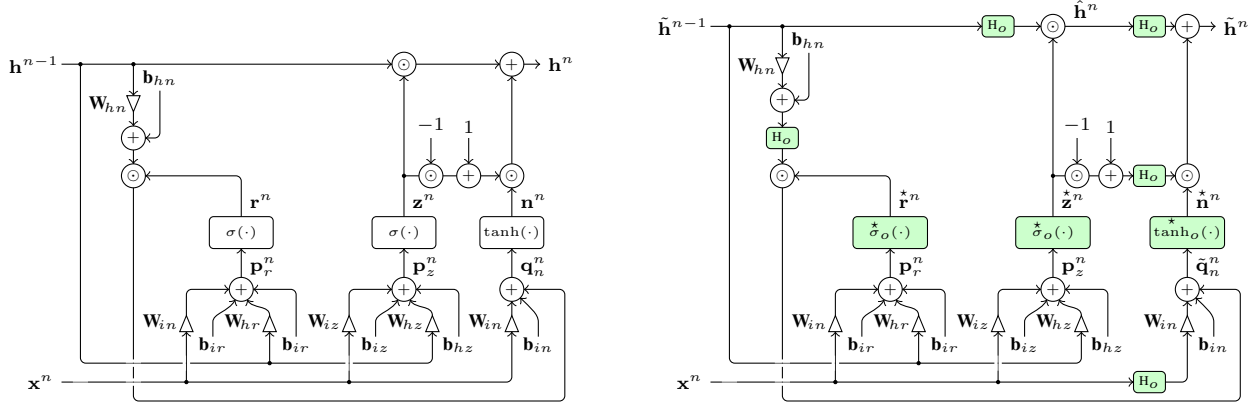
Carson et al. [1] show, based on earlier work from Chowdhury [21], that the system can be run at an oversampled rate

$$f'_{\mathrm{SR}} = M f_{\mathrm{SR}}, \tag{14}$$

where $\{M \in \mathbb{R}^+ \mid M \geq 1\}$ is the oversampling (OS) factor, f_{SR} the SR, and $'$ is used to denote a quantity at the higher rate. This is achieved by delaying the state recursion such that the continuous-time delay matches that seen during training, or

$$\mathbf{h}'^n = f(\mathbf{x}'^n, \mathbf{h}'^{n-M}). \tag{15}$$

In cases where $M \in \mathbb{Z}^+$, the required past states are known and require no approximation, resulting in high-quality oversampled operation. In the general case of $M \in \mathbb{R}^+$, the required past states can be approximated by using a fractional delay filter, resulting in modifications in model tonality. The corresponding network architecture is visualized in Fig. 1.



(a) *Original* (b) *Adjusted*

Figure 2: *Original and adjusted block diagram for the GRU, with added or modified elements highlighted in green.*

3. METHOD

Our method operates on the following assumptions:

1. The underlying recurrent unit is of an explicit form.
2. The used activations have tractable antiderivatives.

The method consists of the following steps:

1. Start with an RNN fulfilling the requirements from above.
2. Treat the activation functions:
   (a) Solve for the antiderivatives of the activation functions up to the required order o.
   (b) Replace each activation function with its antialiased counterpart using ADAA.
3. Trace the forward paths:
   (a) Identify locations where the antialiased activations cause misalignment of the internal signals.
   (b) Add a synchronization filter representing the linearization of the ADAA method to the required paths.
4. Adjust SR:
   (a) Compute the total group delay caused by the antialiased activations and synchronization filters.
   (b) (Optional) Calculate an additional oversampling term to reach a desired SR.
   (c) Apply the SR independent RNNs technique to adjust the model SR.

In the following subsections, we show in detail how the method can be applied to the GRU (Sec. 3.1) and the LSTM (Sec. 3.2).

3.1. Gated Recurrent Unit (GRU)

Recall the governing equations for the GRU, shown as a block diagram in Fig. 2a, following the notation used in *PyTorch*:

$$\mathbf{r}^n = \sigma(\mathbf{W}_{ir}\mathbf{x}^n + \mathbf{b}_{ir} + \mathbf{W}_{hr}\mathbf{h}^{n-1} + \mathbf{b}_{hr}) \tag{16}$$

$$\mathbf{z}^n = \sigma(\mathbf{W}_{iz}\mathbf{x}^n + \mathbf{b}_{iz} + \mathbf{W}_{hz}\mathbf{h}^{n-1} + \mathbf{b}_{hz}) \tag{17}$$

$$\mathbf{n}^n = \tanh(\mathbf{W}_{in}\mathbf{x}^n + \mathbf{b}_{in} + \mathbf{r}^n \odot (\mathbf{W}_{hn}\mathbf{h}^{n-1} + \mathbf{b}_{hn})) \tag{18}$$

$$\mathbf{h}^n = (1 - \mathbf{z}^n) \odot \mathbf{n}^n + \mathbf{z}^n \odot \mathbf{h}^{n-1}, \tag{19}$$

where $(\mathbf{r}^n, \mathbf{z}^n, \mathbf{n}^n) \in \mathbb{R}^{\mathrm{H}}$ are the reset, update and new gates, respectively, $\mathbf{W}_* \in \mathbb{R}^{\mathrm{X}\times\mathrm{H}}$ and $\mathbf{b}_* \in \mathbb{R}^{\mathrm{H}}$ are the weights and biases, $\odot$ is the Hadamard product, and $(\sigma, \tanh) : \mathbb{R}^{\mathrm{X}} \times \mathbb{R}^{\mathrm{H}} \mapsto \mathbb{R}^{\mathrm{H}}$ are the sigmoid function and hyperbolic tangent, commonly called *activation functions* in machine learning context. Note that both the Hadamard product and the (memoryless) activations inherently expand the spectrum, making them susceptible to aliasing.

The bandwidth extension caused by the Hadamard product is at worst double that of its inputs— this would be the case of two sinusoids at $f_{\mathrm{SR}}/2$ combining to form a sum frequency at f_{SR}. The activations, however, can expand the bandwidth an infinite amount (albeit typically with some "roll-off"). As result, this work focuses solely on antialiasing the activations and leaves explicit consideration of the Hadamard product out of scope.

Let's denote the arguments to the activations as

$$\mathbf{p}_r^n = \mathbf{W}_{ir}\mathbf{x}^n + \mathbf{b}_{ir} + \mathbf{W}_{hr}\mathbf{h}^{n-1} + \mathbf{b}_{hr} \tag{20}$$

$$\mathbf{p}_z^n = \mathbf{W}_{iz}\mathbf{x}^n + \mathbf{b}_{iz} + \mathbf{W}_{hz}\mathbf{h}^{n-1} + \mathbf{b}_{hz} \tag{21}$$

$$\mathbf{q}_n^n = \mathbf{W}_{in}\mathbf{x}^n + \mathbf{b}_{in} + \mathbf{r}^n \odot (\mathbf{W}_{hn}\mathbf{h}^{n-1} + \mathbf{b}_{hn}), \tag{22}$$

where $(\mathbf{p}_r^n, \mathbf{p}_z^n, \mathbf{q}_n^n) : \mathbb{R}^{\mathrm{X}} \times \mathbb{R}^{\mathrm{H}} \mapsto \mathbb{R}^{\mathrm{H}}$. Plugging into Eqs. (16)–(18) leads to

$$\mathbf{r}^n = \sigma(\mathbf{p}_r^n) : \mathbb{R}^{\mathrm{H}} \mapsto \mathbb{R}^{\mathrm{H}} \tag{23}$$

$$\mathbf{z}^n = \sigma(\mathbf{p}_z^n) : \mathbb{R}^{\mathrm{H}} \mapsto \mathbb{R}^{\mathrm{H}} \tag{24}$$

$$\mathbf{n}^n = \tanh(\mathbf{q}_n^n) : \mathbb{R}^{\mathrm{H}} \mapsto \mathbb{R}^{\mathrm{H}}, \tag{25}$$

clearly revealing that, in effect, the functions' arguments are simply vectors in $\mathbb{R}^{\mathrm{H}}$. This, together with the channels being treated independently, allows the channel-wise application of ADAA.

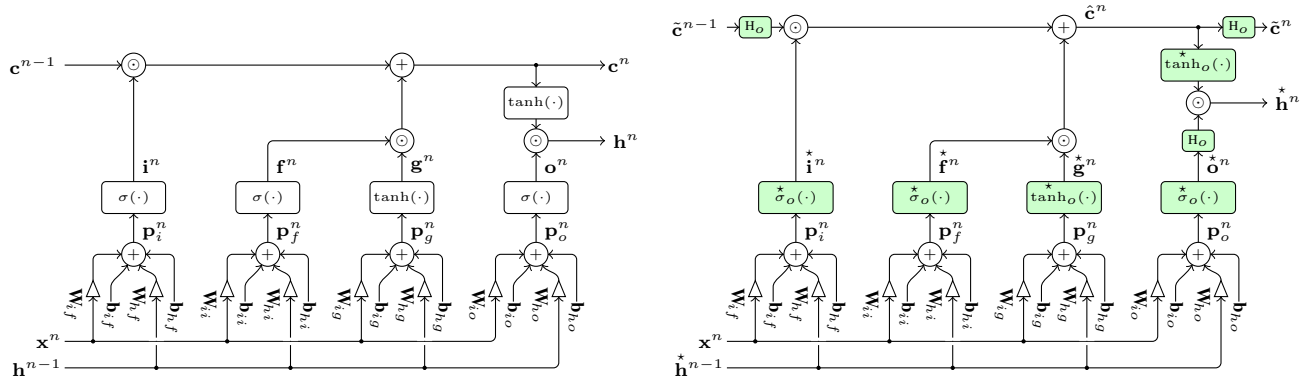
Applying antialiasing to Eqs. (23) & (24) leads to

$$\overset{\star}{\mathbf{r}}{}^n = \overset{\star}{\sigma}_o(\mathbf{p}_r^n, ...) \tag{26}$$

$$\overset{\star}{\mathbf{z}}{}^n = \overset{\star}{\sigma}_o(\mathbf{p}_z^n, ...), \tag{27}$$

which requires evaluating the antiderivatives of σ up to the oth order, with examples given later in Sec. 4. In order to apply antialiasing to Eq. (25), we must first align the internal signals in $\mathbf{q}_n^n$ similarly as in Sec. 2.1. Using an order-dependent synchronization filter $\mathrm{H}_o(\alpha^n, ...)$ defined earlier, the synchronized $\mathbf{q}_n^n$ becomes

$$\begin{aligned}\tilde{\mathbf{q}}_{\mathbf{n}}^n = \mathbf{W}_{in}\mathrm{H}_o(\mathbf{x}^n, ...) + \mathbf{b}_{in} \\ + \overset{\star}{\mathbf{r}}{}^n \odot (\mathbf{W}_{hn}\mathrm{H}_o(\mathbf{h}^{n-1}, ...) + \mathbf{b}_{hn}),\end{aligned} \tag{28}$$



(a) *Original* (b) *Adjusted*

Figure 3: *Original and adjusted block diagram for the LSTM, with added or modified elements highlighted in green.*

where ~ is used to signify signal synchronization. With this, the antialiased $\mathbf{n}^n$ becomes

$$\overset{\star}{\mathbf{n}}^n = \overset{\star}{\tanh}_o(\tilde{\mathbf{q}}_\mathbf{n}^n, ...), \tag{29}$$

with examples of the required antiderivatives again given later.

Note that each signal in $\overset{\star}{\mathbf{n}}^n$ goes through a cascade of two operators that work on the signals' group delay, as per Eqs. (28) & (29). Thus, the group delay seen by $\overset{\star}{\mathbf{n}}^n$ is twice that of $(\overset{\star}{\mathbf{r}}^n, \overset{\star}{\mathbf{z}}^n)$. With this in mind, to synchronize the signals in Eq. 19, let's first define a helper variable

$$\hat{\mathbf{h}}^n = \overset{\star}{\mathbf{z}}^n \odot \mathrm{H}_o(\mathbf{h}^{n-1}, ...), \tag{30}$$

where $\hat{}$ is used to denote an intermediate signal, to align $\mathbf{h}^{n-1}$ with $\overset{\star}{\mathbf{z}}^n$. Keeping track of the individual group delays, the synchronized $\mathbf{h}^n$ becomes

$$\tilde{\mathbf{h}}^n = (\mathbf{1} - \mathrm{H}_o(\overset{\star}{\mathbf{z}}^n, ...)) \odot \overset{\star}{\mathbf{n}}^n + \mathrm{H}_o(\hat{\mathbf{h}}^n, ...). \tag{31}$$

Collecting the equations above, the final set of equations for the antialiased GRU becomes:

$$\overset{\star}{\mathbf{r}}^n = \overset{\star}{\sigma}_o(\mathbf{p}_r^n, ...) \tag{32}$$
$$\overset{\star}{\mathbf{z}}^n = \overset{\star}{\sigma}_o(\mathbf{p}_z^n, ...) \tag{33}$$
$$\overset{\star}{\mathbf{n}}^n = \overset{\star}{\tanh}_o(\tilde{\mathbf{q}}_\mathbf{n}^n, ...) \tag{34}$$
$$\hat{\mathbf{h}}^n = \overset{\star}{\mathbf{z}}^n \odot \mathrm{H}_o(\tilde{\mathbf{h}}^{n-1}, ...) \tag{35}$$
$$\tilde{\mathbf{h}}^n = (1 - \mathrm{H}_o(\overset{\star}{\mathbf{z}}^n, ...)) \odot \overset{\star}{\mathbf{n}}^n + \mathrm{H}_o(\hat{\mathbf{h}}^n, ...), \tag{36}$$

where $(\mathbf{p}_r^n, \mathbf{p}_z^n, \tilde{\mathbf{q}}_\mathbf{n}^n)$ are as in Eqs. (20), (21) & (28), and the sequence $\mathbf{h}$ replaced with $\tilde{\mathbf{h}}$ throughout. Due to the increased delay in the state update recursion, the system is now native to an oversampled rate $f'_{\mathrm{SR}} = (o+1)f_{\mathrm{SR}}$, per Sec. 2.2. The block diagram corresponding to the set of equations is shown in Fig. 2b.

3.2. Long Short-Term Memory (LSTM)

The derivation for the LSTM is similar to the GRU. Following again the *PyTorch* notation, recall the LSTM governing equations listed below and shown as a block diagram in Fig. 3a:

$$\mathbf{i}^n = \sigma(\mathbf{W}_{ii}\mathbf{x}^n + \mathbf{b}_{ii} + \mathbf{W}_{hi}\mathbf{h}^{n-1} + \mathbf{b}_{hi}) \tag{37}$$
$$\mathbf{f}^n = \sigma(\mathbf{W}_{if}\mathbf{x}^n + \mathbf{b}_{if} + \mathbf{W}_{hf}\mathbf{h}^{n-1} + \mathbf{b}_{hf}) \tag{38}$$
$$\mathbf{g}^n = \tanh(\mathbf{W}_{ig}\mathbf{x}^n + \mathbf{b}_{ig} + \mathbf{W}_{hg}\mathbf{h}^{n-1} + \mathbf{b}_{hg}) \tag{39}$$
$$\mathbf{o}^n = \sigma(\mathbf{W}_{io}\mathbf{x}^n + \mathbf{b}_{io} + \mathbf{W}_{ho}\mathbf{h}^{n-1} + \mathbf{b}_{ho}) \tag{40}$$
$$\mathbf{c}^n = \mathbf{f}^n \odot \mathbf{c}^{n-1} + \mathbf{i}^n \odot \mathbf{g}^n \tag{41}$$
$$\mathbf{h}^n = \mathbf{o}^n \odot \tanh(\mathbf{c}^n), \tag{42}$$

where $(\mathbf{i}^n, \mathbf{f}^n, \mathbf{g}^n, \mathbf{o}^n) \in \mathbb{R}^\mathrm{H}$ are the current input, forget, cell and output gates, respectively, $\mathbf{c}^n \in \mathbb{R}^\mathrm{H}$ the current cell state, and the rest as in Sec. 3.1. Notice the same bandwidth expanding operators as in the GRU, as well as the two distinct state vectors $(\mathbf{c}^n, \mathbf{h}^n)$.

As before, let's denote the arguments to the activations as

$$\mathbf{p}_i^n = \mathbf{W}_{ii}\mathbf{x}^n + \mathbf{b}_{ii} + \mathbf{W}_{hi}\mathbf{h}^{n-1} + \mathbf{b}_{hi} \tag{43}$$
$$\mathbf{p}_f^n = \mathbf{W}_{if}\mathbf{x}^n + \mathbf{b}_{if} + \mathbf{W}_{hf}\mathbf{h}^{n-1} + \mathbf{b}_{hf} \tag{44}$$
$$\mathbf{p}_g^n = \mathbf{W}_{ig}\mathbf{x}^n + \mathbf{b}_{ig} + \mathbf{W}_{hg}\mathbf{h}^{n-1} + \mathbf{b}_{hg} \tag{45}$$
$$\mathbf{p}_o^n = \mathbf{W}_{io}\mathbf{x}^n + \mathbf{b}_{io} + \mathbf{W}_{ho}\mathbf{h}^{n-1} + \mathbf{b}_{ho}, \tag{46}$$

where $(\mathbf{p}_i^n, \mathbf{p}_f^n, \mathbf{p}_g^n, \mathbf{p}_o^n) : \mathbb{R}^\mathrm{X} \times \mathbb{R}^\mathrm{H} \mapsto \mathbb{R}^\mathrm{H}$. Inserting into Eqs. (37)–(40) produces

$$\mathbf{i}^n = \sigma(\mathbf{p}_i^n) : \mathbb{R}^\mathrm{H} \mapsto \mathbb{R}^\mathrm{H} \tag{47}$$
$$\mathbf{f}^n = \sigma(\mathbf{p}_f^n) : \mathbb{R}^\mathrm{H} \mapsto \mathbb{R}^\mathrm{H} \tag{48}$$
$$\mathbf{g}^n = \tanh(\mathbf{p}_g^n) : \mathbb{R}^\mathrm{H} \mapsto \mathbb{R}^\mathrm{H} \tag{49}$$
$$\mathbf{o}^n = \sigma(\mathbf{p}_o^n) : \mathbb{R}^\mathrm{H} \mapsto \mathbb{R}^\mathrm{H}, \tag{50}$$

where the function domain is again reduced to $\mathbb{R}^\mathrm{H}$, leading directly to the antialiased variants

$$\overset{\star}{\mathbf{i}}^n = \overset{\star}{\sigma}_o(\mathbf{p}_i^n, ...) \tag{51}$$
$$\overset{\star}{\mathbf{f}}^n = \overset{\star}{\sigma}_o(\mathbf{p}_f^n, ...) \tag{52}$$
$$\overset{\star}{\mathbf{g}}^n = \overset{\star}{\tanh}_o(\mathbf{p}_g^n, ...) \tag{53}$$
$$\overset{\star}{\mathbf{o}}^n = \overset{\star}{\sigma}_o(\mathbf{p}_o^n, ...). \tag{54}$$

To apply antialiasing to the states, we begin by aligning the signals in Eq. (41) by defining a helper variable

$$\hat{\mathbf{c}}^n = \overset{\star}{\mathbf{f}}^n \odot \mathrm{H}_o(\mathbf{c}^{n-1}, ...) + \overset{\star}{\mathbf{i}}^n \odot \overset{\star}{\mathbf{g}}^n. \tag{55}$$

With this, the antialiased hidden state becomes

$$\overset{\star}{\mathbf{h}}^n = \mathrm{H}_o(\overset{\star}{\mathbf{o}}^n, ...) \odot \overset{\star}{\tanh}_o(\hat{\mathbf{c}}^n, ...), \tag{56}$$

where again we've doubled the group delay similarly as before. Finally, we synchronize the intermediate cell state $\hat{c}$ by

$$\tilde{\mathbf{c}}^n = \mathrm{H}_o(\hat{\mathbf{c}}^n, ...). \tag{57}$$

Table 2: *Antiderivatives for activations.*

Activation	Order	Antiderivative
σ	1	$\log(\mathrm{e}^x + 1)$
	2	$-\mathrm{Li}_2(-\mathrm{e}^x) + \pi^2/12$
$\tanh$	1	$\log(\cosh(x))$
	2	$\frac{1}{2}(\mathrm{Li}_2(-\mathrm{e}^{-2x}) - x(x + 2\log(\mathrm{e}^{-2x} + 1))$ $+2\log(\cosh(x))) + \pi^2/24$

Table 3: *Model candidates.*

Rec. Unit	Order	M_{OS}	M	f_{SR}(kHz)
GRU/LSTM	—	1	1	44.1
	—	2	2	88.2
	—	4	4	176.4
ADAAGRU/LSTM	1	1	2	88.2
	1	3	4	176.4
	2	2	4	176.4

Collecting from above, the final set of equations for the antialiased LSTM becomes:

$$\overset{\star}{\mathbf{i}}^n = \overset{\star}{\sigma}_o(\mathbf{p}_i^n, ...) \tag{58}$$
$$\overset{\star}{\mathbf{f}}^n = \overset{\star}{\sigma}_o(\mathbf{p}_f^n, ...) \tag{59}$$
$$\overset{\star}{\mathbf{g}}^n = \overset{\star}{\tanh}_o(\mathbf{p}_g^n, ...) \tag{60}$$
$$\overset{\star}{\mathbf{o}}^n = \overset{\star}{\sigma}_o(\mathbf{p}_o^n, ...) \tag{61}$$
$$\hat{\mathbf{c}}^n = \overset{\star}{\mathbf{f}}^n \odot \mathrm{H}_o(\tilde{\mathbf{c}}^{n-1}, ...) + \overset{\star}{\mathbf{i}}^n \odot \overset{\star}{\mathbf{g}}^n \tag{62}$$
$$\overset{\star}{\mathbf{h}}^n = \mathrm{H}_o(\overset{\star}{\mathbf{o}}^n, ...) \odot \overset{\star}{\tanh}_o(\hat{\mathbf{c}}^n, ...) \tag{63}$$
$$\tilde{\mathbf{c}}^n = \mathrm{H}_o(\hat{\mathbf{c}}^n, ...), \tag{64}$$

where $(\mathbf{p}_i^n, \mathbf{p}_f^n, \mathbf{p}_g^n, \mathbf{p}_o^n)$ are as in Eqs. (43)–(46) and the series $(\mathbf{c}, \mathbf{h})$ have been replaced by $(\tilde{\mathbf{c}}, \overset{\star}{\mathbf{h}})$ throughout. Similarly as for the GRU, the system is native to an oversampled rate $f'_{\text{SR}} = (o+1)$ f_{SR} due to the increased delay in the state update. The corresponding block diagram is shown in Fig. 3b.

3.3. Combining with SR Independent RNNs

To allow for SR independent operation, the proposed technique is combined with the SR independent RNNs method introduced in Sec. 2.2. Following the earlier notation, let's split the total OS factor M to that resulting from ADAA being applied $M_{\text{ADAA}} = (o+1)$, as well as an additional OS term M_{OS} required to run the system at a desired rate f'_{SR}. The total OS factor is

$$\begin{aligned} M &= M_{\text{ADAA}} + M_{\text{OS}} - 1 \\ &= (o+1) + M_{\text{OS}} - 1 \\ &= o + M_{\text{OS}}, \end{aligned} \tag{65}$$

giving a combined SR independent and antialiased state update

$$\mathbf{h'}^n = \overset{\star}{f}_o(\mathbf{x'}^n, \mathbf{h'}^{n-(o+M_{\text{OS}})}). \tag{66}$$

4. EXPERIMENTS

This section provides an overview of the experimental procedure, including the choice of models, the evaluation strategy and remarks concerning the implementation.

4.1. Models and Baselines

Pre-trained GRU and LSTM models were used for testing the proposed algorithms. Models trained on guitar amplifier data were chosen due to the target devices' highly nonlinear behavior.

For the GRU, the Blackstar HT-1 model trained by Wright et al. [3] was used. The network consists of a size 32 GRU with a single recurrent layer, a FC output layer without an activation, and a skip connection linking the network input to its output. For the LSTM, those available in the *GuitarML Tone Library*[2] *Proteus Tone Pack* were considered, and the Mesa Mini Rectifier model was chosen for detailed analysis due to its high gain. Three additional LSTM models from the pack were added as supplementary material into the website[1]. All models in the pack are otherwise similar to the considered GRU model, but utilize a size 40 LSTM as the recurrent unit. All networks were originally trained at a base rate f_{SR} of 44.1 kHz.

1st and 2nd-order variants of the proposed algorithm were tested for both models. The required antiderivatives for evaluating the 1st and 2nd-order activations are listed in Tab. 2 where Li_2 is the dilogarithm / Spence's function [22], $\log$ the natural logarithm and $\cosh$ the hyperbolic cosine. The filters required for synchronizing the internal signals can be found from Tab. 1. In the 1st-order case, applying the method results in $M = 2\times$ OS factor in comparison to the model base rate. In the 2nd-order case, the resulting OS factor is $3\times$. To test the algorithm at commonly used OS factors which are powers of two, the OS factors resulting from applying the algorithm were augmented with an additional term M_{OS}, leading to the final considered candidates in Tab. 3.

In order to compare the algorithms against existing methods, high and low quality baselines were included in the test. For a high quality baseline, the SR independent RNNs technique (Sec. 2.2) was used, which was considered the state-of-the-art. For a low-quality anchor, a regular RNN without oversampling was chosen.

4.2. Evaluation and Data

We adapt an evaluation strategy consisting of objective metrics as well as subjective assessment used in earlier work [1, 8, 19, 9].

The objective evaluation follows that used by Carson et al. [1]. The procedure consists of exciting the models under study with pure sine tones over the range produced by a standard piano from A_0 to C_8 and computing the signal to aliasing noise ratio (SNRA) from the resulting outputs. For the models considered in this study, static sine tones were not used during training. Computing the SNRA requires constructing a bandlimited reference signal based on the Discrete Fourier Transform of the raw model output, the details of which can be found in the original work [1]. The corresponding signal chain is shown in Fig. 4.

The SNRA is defined as:

$$\text{SNRA} = \frac{\sum_{k=0}^{\mathrm{N_{FFT}}/2} |\mathbf{Y}_{\text{bl}}[k]|^2}{\sum_{k=0}^{\mathrm{N_{FFT}}/2} (|\mathbf{Y}'[k]| - |\mathbf{Y}_{\text{bl}}[k]|)^2}, \tag{67}$$

where $(\mathbf{Y}_{\text{bl}} \in \mathbb{C}^{\mathrm{N_{FFT}}}, \mathbf{Y}' \in \mathbb{C}^{\mathrm{N'_{FFT}}})$ with $\mathrm{N_{FFT}}' = M\mathrm{N_{FFT}}$ are the spectra of the bandlimited and raw model outputs, respectively,

[2] https://guitarml.com/tonelibrary/tonelib-pro

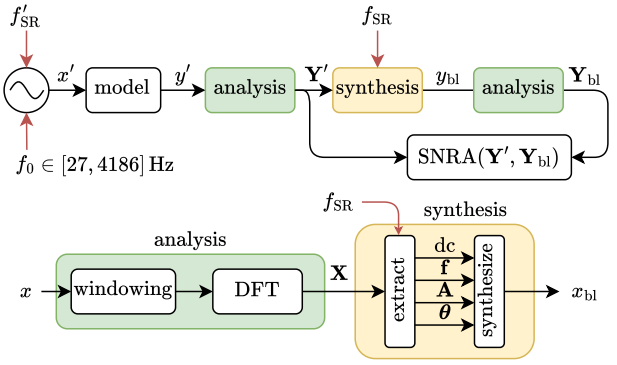


Figure 4: *Signal to aliasing noise ratio (SNRA) computation [1].*

and $|\cdot|$ is the complex modulus. Note that only the harmonic components up to the lower Nyquist frequency $N_{FFT}/2$ are considered.

We find the analysis-resynthesis process sensitive to the level of distortion in the analyzed signal, causing abrupt collapses in the resulting SNRA. To mitigate this, sine amplitudes are set per model to compensate for their distinct gains, effectively preventing the collapse. Suitable levels were determined heuristically by decrementing the sine amplitudes in 6 dB steps from 0 dBFS until no modal collapse was encountered. The final selected gains were $(-24\,\text{dBFS}, -12\,\text{dBFS})$ for the GRU and LSTM models, respectively. We hypothesize that this finding may explain the abrupt drops observed in related figures in the original paper [1].

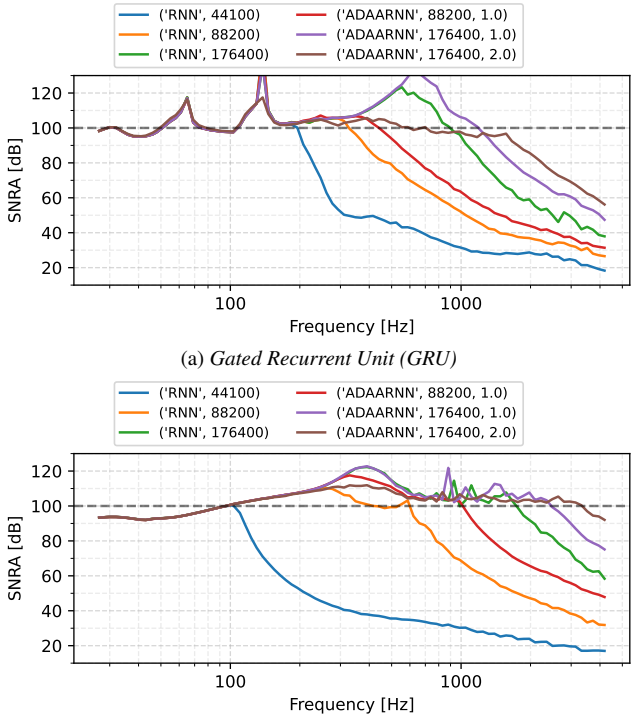
We experiment with the signal to harmonic noise ratio (SNRH) metric proposed by Carson et al. [1] to objectively assess deviations in non-aliased harmonics, but encounter difficulties in producing interpretable graphs suitable for analyzing the algorithms' behavior. To address this, the harmonic deviation is evaluated using individual sines from the previously used series of frequencies, comparing the magnitudes of the non-aliased harmonic overtones produced by the model variants, following prior work [19, 9]. While this sacrifices some generality, we believe the deeper insight gained from detailed observations outweighs this limitation.

Finally, as a supplementary subjective measure, the models were excited with linear sine sweeps from 1 Hz to 20 kHz with the gains set to those used earlier. Due to space constraints, the results are made available in the accompanying website[1].

4.3. Implementation Details

PyTorch was used across all experiments. The 1st and 2nd-order antialiased forward passes for the activations were implemented following Bilbao's alternative derivation, which can be found written out explicitly in earlier work [20, 5]. The dilogarithm was evaluated using the implementation available in *SciPy*. To prevent numerical instabilities, computations were performed in double-precision floating-point arithmetic with an epsilon term $\epsilon = 10^{-3}$.

The analysis-resynthesis chain used for computing the SNRA was implemented following Carson et al. [1]. Importantly, double-precision floating-point arithmetic was used throughout, and the sine fundamentals truncated to the nearest integer to improve the dynamic range of the resulting graphs. This truncation resulted in fundamental frequencies $f_0 \in [27, 4186]$Hz within the considered range. The signal duration was set to 2 s, letting the model warm up for 1 s and using the remaining 1 s for analysis.



(a) *Gated Recurrent Unit (GRU)*

(b) *Long Short-Term Memory (LSTM)*

Figure 5: *Signal to aliasing noise (SNRA) results.*

5. RESULTS

This section is for discussing the results. For additional content, see the accompanying webpage[1].

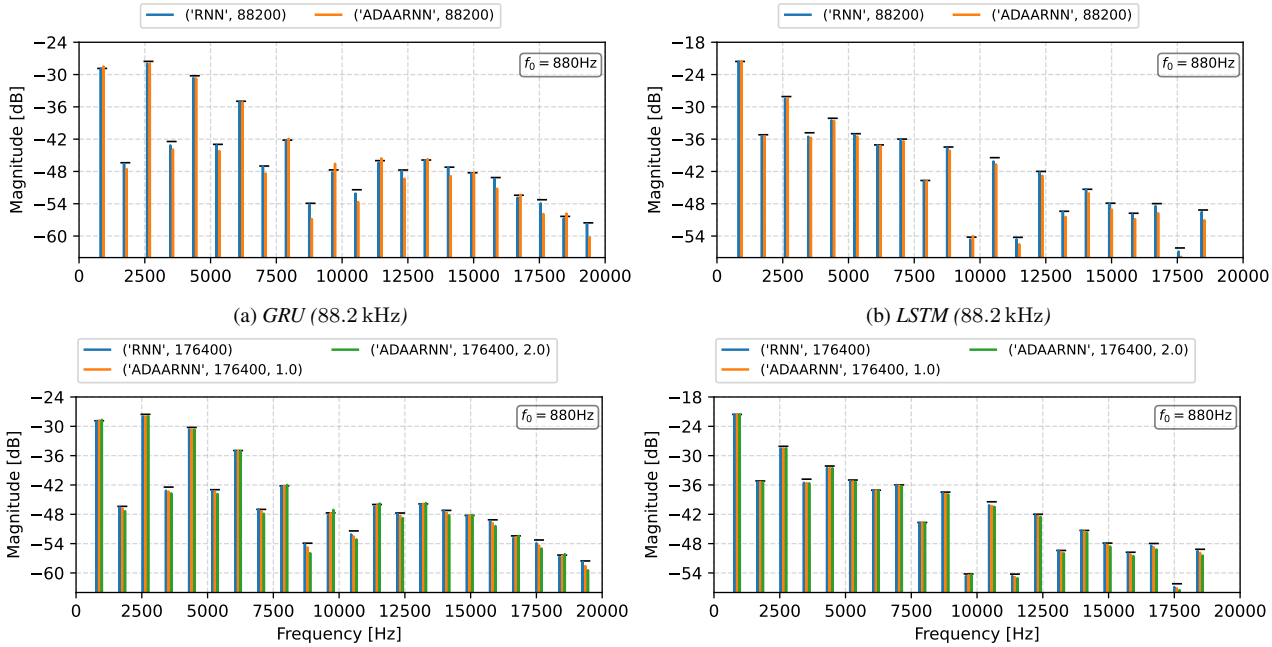
5.1. Alias Rejection

Fig. 5 presents the SNRA analysis results for the different model variants and considered SRs, shown on log-scale as a function of the sine fundamental frequency. Across all models, the SNRA values follow a similar trend, remaining high ($\geq \approx 100\,\text{dB}$) for a proportion of the low-to-mid frequencies before dropping due to aliasing. As expected, increasing the OS factor reduces aliasing for both considered model types. Crucially, for the considered SRs $f'_{SR} \in (88.2, 176.4)$kHz where the proposed ADAA technique is applicable, applying the method improves the SNRA compared to the corresponding baseline models. Furthermore, increasing the ADAA order from 1st to 2nd further enhances the performance, as expected from theory.

Comparing the results for the GRU and LSTM models, the SNRA values are generally lower for the former, indicating a higher level of aliasing. This can be attributed to the greater gain exhibited by the underlying model, clarifying the need for more gain reduction to keep the SNRA from collapsing, as discussed in Sec. 4.2.

For the GRU models at 88.2 kHz (Fig. 5a), applying 1st-order ADAA results in an SNRA improvement of ≈20 dB over the baseline model when the values begin to drop from 100 dB, with the margin decreasing towards high frequencies. At 176.4 kHz, where both 1st and 2nd-order methods are applicable, the improvements over the baseline start at approximately ≈20 dB and ≈30 dB for the respective orders, again diminishing towards high frequencies.

The trends for the LSTM models (Fig. 5b) are largely sim-



(a) *GRU* (88.2 kHz) (b) *LSTM* (88.2 kHz)

(c) *GRU* (176.4 kHz) (d) *LSTM* (176.4 kHz)

Figure 6: *Magnitudes of non-aliased harmonics generated by the considered models using* 880 Hz *sinusoidal excitation. Horizontal lines mark baseline model harmonics, with candidate model harmonics offset for clarity.*

ilar, although the proposed antialiasing scheme produces larger improvements over their baseline counterparts in comparison to earlier. At 88.2 kHz, applying 1st-order ADAA results in an initial SNRA improvement of approximately ≈30 dB over the baseline, with the margin decreasing towards high frequencies similar to earlier. At 176.4 kHz, the improvements start at ≈20 dB and ≈40 dB for the 1st and 2nd-order methods, respectively, before gradually decreasing with frequency.

5.2. Preservation of Harmonics

Fig. 6 presents the magnitudes of the non-aliased harmonic overtones produces by the different model variants at the evaluated SRs, using a 880 Hz sinusoidal input with gains matching those used earlier. The harmonics generated by the baseline model at 44.1 kHz are treated as ground truth and are marked in each subfigure with horizontal lines. The harmonics of the candidate models are offset horizontally for visualization purposes, and the dynamic range adjusted to ensure visibility of harmonics within the nominal hearing range $f \in [20, 20\text{k}]$Hz.

Generally, and as expected from earlier findings [1], oversampling the baseline RNN models by integer factors $M \in Z^+$ retains the locations of non-aliased harmonics to a good accuracy, although minor deviations can be observed for some of the even harmonics. Applying the proposed antialiasing technique preserves the overall harmonic pattern but introduces more pronounced deviations into the harmonics' magnitudes. These deviations follow distinct trends for the odd and even harmonic series and vary between the considered GRU and LSTM models.

Starting from the 1st-order antialiased GRU model at 88.2 kHz (Fig. 6a), the louder odd harmonic series follows the ground truth closely, with slight amplification observed around ≈10 kHz and towards 20 kHz. The quieter even series is slightly attenuated across the frequency range, with more pronounced attenuation occurring in the same regions where the odd series showed larger deviations. We attribute the increased attenuation towards 20 kHz to the well-known low-pass effect of ADAA [8], though it remains unclear why the odd harmonic series seems unaffected.

The GRU models at 176.4 kHz (Fig. 6c) exhibit similar trends, although with smaller deviations in comparison to earlier. The additionally 3× oversampled 1st-order model performs notably better across the frequency range, with harmonic overtones aligning more closely with the ground truth. For the 2nd-order model, the deviations resemble those of the 1st-order model at 88.2 kHz, although being generally less pronounced. Overall, the additionally 3× oversampled 1st-order model demonstrates the best performance among the evaluated antialiased GRU models.

For the LSTM models, the deviations are in general less pronounced, and they share some common traits with the analyzed GRU models. Examining the 1st-order antialiased LSTM model at 88.2 kHz (Fig. 6b), the odd harmonic series closely follows that of the ground truth until around ≈10 kHz, after which a low-pass effect can be observed. This suggests that the amplification observed in the corresponding GRU model extends beyond the readily visible region around ≈10 kHz, masking the expected low-pass effect in previous analysis. The results for the even harmonic series are largerly similar, with minor deviations in some of the low-order harmonics (which are also observed for the baseline RNN), and a low-pass effect similar to the odd series towards 20 kHz.

Observing the LSTM model behavior at 176.4 kHz (Fig. 6d), the deviations become less pronounced similarly as for the GRU models. Similarly as before, the performance of the additionally 3× oversampled 1st-order model is improved considerably, and the 2nd-order model performance landing in between the 1st order models at 88.2 and 176.4 kHz. As before, the additionally oversampled 1st-order model demonstrates the best overall performance among the evaluated antialiased LSTM model candidates.

6. CONCLUSIONS

This work proposed an extension of the ADAA method to explicit, computable, RNNs, treating the GRU and LSTM as case studies. These recurrent units are widely used in general sequence modeling tasks, such as those in VA modeling. The extension was evaluated in detail on two pre-trained guitar amplifier models—the Blackstar HT-1 and the Mesa Mini Rectifier—with results for additional model candidates made available in the accompanying webpage[1]. Algorithm behavior was assessed in terms of both alias rejection and preservation of non-aliased harmonics.

The proposed method was found to considerably reduce the models' tendency to alias compared to the current state-of-the-art, as shown by the SNRA analysis results. Increasing the ADAA order further extended the margin to the baseline, consistent with the underlying theory. A trade-off was observed between alias rejection and the preservation of model tonality. In addition to the expected low-pass effect due to ADAA, further deviations in both even and odd harmonic series were encountered. For the models studied, the additionally $3\times$ oversampled 1st-order model running at $4\times$ the original SR presented a good compromise between alias rejection, harmonic preservation, and system complexity. Ultimately, the choice of order is closely linked to the target system, and should ideally be considered on a per-system basis.

Given that the proposed operations are inherently differentiable, they could also be applied during training. We hypothesize this could allow the model to learn to compensate for harmonic deviations, mitigating some of the issues noted earlier. Related work in image processing has demonstrated the critical role of alias rejection during training, demonstrating how incorporated antialiasing measures can improve model performance and lead to state-of-the-art results [23]. However, our approach does not *require* retraining, unlike the antialiased STN approach [13].

Future work should explore the computational costs incurred by the method at varying sample rates and hidden sizes to better understand its relation to the oversampling approach. Additionally, the investigation of alternative synchronization filters and higher-order antialiasing filter kernels [24] should be conducted, potentially providing insight into the cause of the disturbed harmonic overtones. Further, explicit consideration of the bandwidth expansion of the Hadamard products should be pursued.

7. ACKNOWLEDGMENTS

Thank you Ken Bogdanowicz for supporting this research and for providing the resources, as well as Chris Santoro and Eugene Umlor for the valuable discussions throughout the project.

8. REFERENCES

[1] A. Carson, A. Wright, J. Chowdhury, V. Välimäki, and S. Bilbao, "Sample rate independent recurrent neural networks for audio effects processing," in *Proc. 27th Int. Conf. Digital Audio Effects (DAFx24)*, Guildford, UK, Sept. 2024, pp. 17–24.

[2] C. J. Steinmetz and J. D. Reiss, "Efficient neural networks for real-time modeling of analog dynamic range compression," in *Proc. Audio Eng. Soc. 152nd Conv.*, The Hague, Netherlands, May 2022, Audio Engineering Society.

[3] A. Wright, E.-P. Damskägg, and V. Välimäki, "Real-time black-box modelling with recurrent neural networks," in *Proc. 22nd Int. Conf. Digital Audio Effects (DAFx-2019)*, Birmingham, UK, Sept. 2019, pp. 173–180.

[4] J. D. Parker, F. Esqueda, and A. Bergner, "Modelling of nonlinear state-space systems using a deep neural network," in *Proc. 22nd Int. Conf. Digital Audio Effects (DAFx-2019)*, Birmingham, UK, Sept. 2019, pp. 165–172.

[5] F. Esqueda, *Aliasing Reduction in Nonlinear Audio Signal Processing*, Ph.D. thesis, Aalto U., Espoo, Finland, May 2018.

[6] V. Välimäki, J. Pekonen, and J. Nam, "Perceptually informed synthesis of bandlimited classical waveforms using integrated polynomial interpolation," *J. Acoust. Soc. Am.*, vol. 131, no. 1, pp. 974–986, Jan. 2012.

[7] F. Esqueda, V. Välimäki, and S. Bilbao, "Rounding corners with BLAMP," in *Proc. 19th Int. Conf. Digital Audio Effects (DAFx-2016)*, Brno, Czech Republic, Sept. 2016, pp. 121–128.

[8] J. D. Parker, V. Zavalishin, and E. Le Bivic, "Reducing the aliasing of nonlinear waveshaping using continuous-time convolution," in *Proc. 19th Int. Conf. Digital Audio Effects (DAFx-2016)*, Brno, Czech Republic, Sept. 2016, pp. 137–144.

[9] M. Holters, "Antiderivative antialiasing for stateful systems," *Appl. Sci.*, vol. 10, no. 1, Dec. 2019.

[10] K. J. Werner and E. Azelborn, "Antialiasing piecewise polynomial waveshapers," in *Proc. 26th Int. Conf. Digital Audio Effects (DAFx23)*, Copenhagen, Denmark, Sept. 2023, pp. 180–187.

[11] A. Wright, E.-P. Damskägg, L. Juvela, and V. Välimäki, "Real-time guitar amplifier emulation with deep learning," *Appl. Sci.*, vol. 10, no. 3, Jan. 2020.

[12] T. Vanhatalo, P. Legrand, M. Desainte-Catherine, P. Hanna, and G. Pille, "Evaluation of real-time aliasing reduction methods in neural networks for nonlinear audio effects modelling," *J. Audio Eng. Soc.*, vol. 72, no. 3, pp. 114–122, Mar. 2024.

[13] L. Köper and M. Holters, "Antialiased state trajectory neural networks for virtual analog modeling," in *Proc. 26th Int. Conf. Digital Audio Effects (DAFx23)*, Copenhagen, Denmark, Sept. 2023, pp. 165–171.

[14] A. Carson, V. Välimäki, A. Wright, and S. Bilbao, "Resampling filter design for multirate neural audio effect processing," *arXiv:2501.18470*, Jan. 2025.

[15] A. Carson, A. Wright, and S. Bilbao, "Anti-aliasing of neural distortion effects via model fine tuning," *arXiv:2505.11375v1*, May 2025.

[16] R. Sato and J. O. Smith, "Aliasing reduction in neural amp modeling by smoothing activations," *arXiv:2505.04082*, May 2025.

[17] K. Cho et al., "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. 2014 Conf. Empirical Methods Nat. Lang. Process. (EMNLP)*, Doha, Qatar, Oct. 2014, pp. 1724–1734.

[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[19] S. Bilbao, F. Esqueda, J. D. Parker, and V. Välimäki, "Antiderivative antialiasing for memoryless nonlinearities," *IEEE Signal Process. Lett.*, vol. 24, no. 7, pp. 1049–1053, July 2017.

[20] A. Carson, "Aliasing reduction in virtual analogue modelling," M.S. thesis, U. Edinburgh, Edinburgh, UK, Sept. 2020.

[21] J. Chowdhury, "Sample-rate agnostic recurrent neural networks," `https://jatinchowdhury18.medium.com/sample-rate-agnostic-recurrent-neural-networks-238731446b2`, Apr. 2022, accessed 27/06/2023.

[22] L. Lewin, *Dilogarithms and Associated Functions*, MacDonald, London, 1958.

[23] C. Vasconcelos et al., "Impact of aliasing on generalization in deep convolutional networks," in *Proc. 2021 IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Montréal, Canada, Oct. 2021, pp. 10509–10518.

[24] P. P. La Pastina, S. D'Angelo, and L. Gabrielli, "Arbitrary-order IIR antiderivative antialiasing," in *Proc. 24th Int. Conf. Digital Audio Effects (DAFx20in21)*, Vienna, Austria, Sept. 2021, pp. 9–16.