Mikkonen, Otto; Wright, Alec; Välimäki, Vesa

Sampling the user controls in neural modeling of audio devices

Published in:
Eurasip Journal on Audio, Speech, and Music Processing

DOI:
10.1186/s13636-024-00347-5

Published: 01/12/2024

Document Version
Publisher's PDF, also known as Version of record

Published under the following license:
CC BY

Please cite the original version:
Mikkonen, O., Wright, A., & Välimäki, V. (2024). Sampling the user controls in neural modeling of audio devices. *Eurasip Journal on Audio, Speech, and Music Processing*, *2024*(1), Article 26. https://doi.org/10.1186/s13636-024-00347-5

Mikkonen *et al.*
EURASIP Journal on Audio, Speech, and Music Processing (2024) 2024:26
https://doi.org/10.1186/s13636-024-00347-5

EURASIP Journal on Audio, Speech, and Music Processing

EMPIRICAL RESEARCH **Open Access**

Sampling the user controls in neural modeling of audio devices

Otto Mikkonen[1,2*], Alec Wright[1,3] and Vesa Välimäki[1]

Abstract
This work studies neural modeling of nonlinear parametric audio circuits, focusing on how the diversity of settings of the target device user controls seen during training affects network generalization. To study the problem, a large corpus of training datasets is synthetically generated using SPICE simulations of two distinct devices, an analog equalizer and an analog distortion pedal. A proven recurrent neural network architecture is trained using each dataset. The difference in the datasets is in the sampling resolution of the device user controls and in their overall size. Based on objective and subjective evaluation of the trained models, a sampling resolution of five for the device parameters is found to be sufficient to capture the behavior of the target systems for the types of devices considered during the study. This result is desirable, since a dense sampling grid can be impractical to realize in the general case when no automated way of setting the device parameters is available, while collecting large amounts of data using a sparse grid only incurs small additional costs. Thus, the result provides guidance for efficient collection of training data for neural modeling of other similar audio devices.

Keywords Audio systems, Deep learning, Emulation

1 Introduction

Virtual analog (VA) modeling is an active subdiscipline of audio processing that attempts to imitate analog and electromechanical audio hardware using software [1, 2]. Within the past decades, progress in the field has allowed for digital replications of various audio hardware units, including guitar amplifiers [3–5] and synthesizer subcircuits, such as voltage-controlled oscillators [6, 7] and filters [8–10], as well as studio hardware, such as dynamic range compressors [11, 12] and effect processors [13, 14].

The approaches used for VA modeling are traditionally divided into white-box, gray-box, and black-box methods. White-box methods use explicit knowledge of the target circuits to discover and derive the physical constraints that govern the systems, oftentimes encountered as ordinary differential equations (ODEs), which are then discretized and simulated on a computer. Examples of approaches belonging to this category include numerical ODE solvers [3, 15], state-space methods [16], and wave digital filters [5, 9, 17]. Black-box methods use input-output relationships collected from the target circuits together with general-purpose digital models to try and match the observed behavior via optimization. Examples of black-box approaches include block-based methods [18], Volterra series expansion [19], and dynamic convolution [20]. In gray-box modeling methods, a combination of these two approaches is used [12, 21, 22].

Within the past decade, progress in the field of machine learning (ML), especially in deep learning (DL) [23–25], has ignited an exploration within the VA modeling community to the applicability of ML methods to the task of circuit modeling. As in other domains, DL approaches have been shown to be capable of achieving state-of-the-art accuracy, when applied to a variety of modeling

*Correspondence:
Otto Mikkonen
tot.elektro.nik@gmail.com
[1] Acoustics Laboratory, Department of Information and Communications Engineering, Aalto University, Espoo, Finland
[2] Soundtoys, Inc., Burlington, VT, USA
[3] Acoustics and Audio Group, University of Edinburgh, Edinburgh, UK

targets [26–28]. While early examples of DL-based modeling approaches have explored the use of end-to-end neural networks (NNs), such as convolutional neural networks and recurrent neural networks (RNNs), in a black-box manner [29–31], later work has utilized knowledge of the underlying circuits, steering the approaches more towards the white-box end of circuit modeling [32–34].

In order to be optimized for the VA modeling task, DL methods require a dataset representing the target behavior to be collected. Within the supervised learning paradigm [25], the dataset consists of input-output pairs of audio and the related circuit configurations, e.g., the settings for the user controls, in the case of parametric circuits. While the data can be collected using recordings of the target circuit or via circuit simulation, it has not been clear how the space of user controls of the target should be sampled and exposed to the NNs for them to generalize over all possible configurations. This uncertainty is exemplified in the number of differing practices adopted in the field for collecting training data for the case of parametric circuits [31, 35–37]. More precisely, Wright et al. [31] use a uniform sampling grid with 5 points, Hawley et al. [35] use a uniform sampling grid with 10 or 21 points depending on the target, and both Nercessian et al. [36] and Juvela et al. [37] use random sampling without restrictions on the resolution. Moreover, the question becomes increasingly important for circuits with more than a few user controls, since the space of all possible configurations grows exponentially with the number of parameters.

This paper addresses this gap by synthetically generating a large number of datasets for two distinct nonlinear modeling targets and training and comparing the performance of NNs trained on each of these datasets. The datasets differ in the way the target circuit parameter space is sampled, also taking into account the effect the overall dataset size has on the problem. The data is collected by constructing and running SPICE simulations of the modeling targets, allowing for the collection to happen in a strictly controllable and automated manner. For modeling the circuit behavior, a proven RNN architecture is used, with the models only exposed to input-output pairs of audio and circuit configurations of the targets in a black-box manner. To evaluate the performance of the networks, both an objective evaluation, based on comparison of the loss metrics, and a subjective evaluation, based on a listening test, are given. We point the reader to the accompanying web page for additional materials[1].

The rest of this paper is organized as follows. Section 2 gives a mathematical description of black-box VA modeling formulated as a supervised ML task and also describes the deep NN architecture used for modeling. Section 3 presents a technical overview of the two modeling targets, an analog distortion pedal and an analog equalizer. Section 4 describes the data collection and the sampling of the user controls, forming the main body of the work. Section 5 gives an outline for the training procedure. Sections 6 and 7 reports on the evaluation procedure and the results. Finally, Sect. 8 concludes.

2 Neural modeling of audio circuits

From an ML perspective, VA modeling can be seen as a sequence modeling task usually solved as a supervised learning problem. In supervised learning problems [25], the learning algorithm, usually called the model, is trained using a dataset $\mathbb{D}$ of input-output pairs $\left(\mathbf{u}^{(i)}, \mathbf{y}^{(i)}\right)$ to give predictions $\hat{\mathbf{y}}$ for inputs $\mathbf{u} \notin \mathbb{D}$ not seen during training. In black-box VA modeling, the inputs and the outputs are discrete time representations of audio collected from some target device, and in the case of parametric circuits, the model also receives the device configurations $\boldsymbol{\phi}$ for each pair. In this scenario, the dataset $\mathbb{D}$ is of the form:

$$\mathbb{D} = \left\{ \left(\mathbf{u}^{(i)}, \mathbf{y}^{(i)}, \boldsymbol{\phi}^{(i)} \right) \right\}_{i=1}^{N}, \tag{1}$$

where $\mathbf{u}$ is an input vector of sampled audio, $\mathbf{y}$ is the corresponding output vector, $\boldsymbol{\phi}$ is a vector storing the device configuration, and $N = |\mathbb{D}|$ is the size of the dataset.

Most electronic circuits are stateful systems due to the energy storing elements, like capacitors and inductors, present in them [38]. Thus, in order to understand their behavior given some inputs and the circuit configuration, one needs to also observe the evolution of the circuit output. In discrete time and for parametric circuits, this can be written as a recursion:

$$y[n] = f(y[n-1], u[n], \boldsymbol{\phi}[n]), \tag{2}$$

where $y[n]$ and $y[n-1]$ are the circuit outputs at the current and previous time steps, $u[n]$ and $\boldsymbol{\phi}[n]$ are the input to the circuit as well as its configuration at the current time step, and $f(\cdot)$ performs the mapping between these quantities.

In black-box neural VA modeling approaches, the mapping f is approximated with a neural network $f_{\boldsymbol{\theta}}$, whose weights $\boldsymbol{\theta}$ are optimized to minimize a chosen loss function $\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})$ over the dataset $\mathbb{D}$. The loss function $\mathcal{L}$ is used as a metric to evaluate the discrepancy between the true output $\mathbf{y}$ and the model prediction $\hat{\mathbf{y}}$, and the gradient of the loss w.r.t. the model weights $\nabla_{\boldsymbol{\theta}} \mathcal{L}$ is used to iteratively step the weights towards the optimum.

[1] http://research.spa.aalto.fi/publications/papers/jasm24-neural

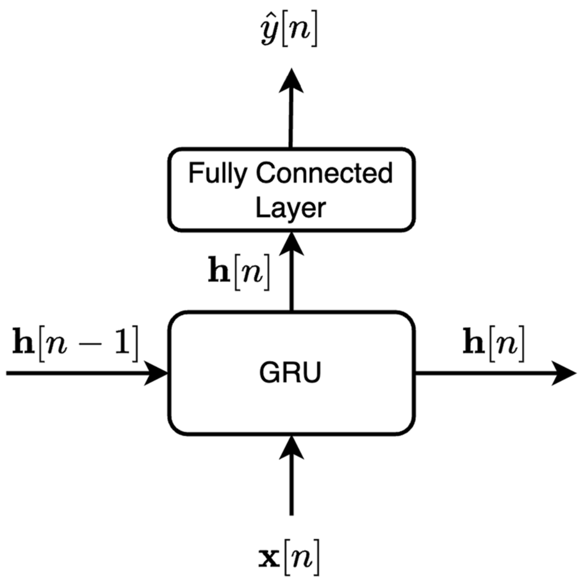

Fig. 1 Deep NN architecture used for the modeling

2.1 Model architecture

Due to the stateful form of the studied systems and following earlier research [29, 31, 33], a stateful NN architecture was chosen for approximating the mapping $f(\cdot)$. In practice, this meant using an RNN, which is a common choice for sequence modeling tasks due to its ability to process inputs of varying lengths [25].

The deep NN architecture used in this study, originally introduced for modeling parametric nonlinear circuits in [31], is shown in Fig. 1. The architecture consists of a gated recurrent unit (GRU) [39] and a fully connected (FC) output layer. The GRU $g_{\theta'}$ computes the state-to-state transition of the model given an input vector $\mathbf{x}[n]$ and its previous hidden state $\mathbf{h}[n-1]$ as follows:

$$\mathbf{h}[n] = g_{\theta'}(\mathbf{x}[n], \mathbf{h}[n-1]). \tag{3}$$

The exact computations performed by a GRU are left out, as they have been described earlier elsewhere [31, 33]. The GRU can be conditioned to adapt to the circuit configurations $\boldsymbol{\phi}[n]$ by concatenating them together with the input audio $u[n]$, making the input vector [31]:

$$\mathbf{x}[n] = [u[n], \boldsymbol{\phi}[n]]. \tag{4}$$

After computing the hidden state $\mathbf{h}[n]$ for the current time step, it is passed on to the FC output layer, which performs a memoryless mapping $\mathbf{h}[n] \to \hat{y}[n]$, producing the output prediction for the current time step. Overall, the model predicts the current output sample as:

$$\hat{y}[n] = f_{\theta}(\mathbf{x}[n], \mathbf{h}[n-1]), \tag{5}$$

where $\mathbf{x}[n]$ is concatenation of the input audio and device configurations as in Eq. (4).

The size of the hidden state $|\mathbf{h}|$, and in turn the input dimensionality of the FC output layer, determines the representational capabilities of the model and is one of its hyperparameters. In our work, a hidden size $|\mathbf{h}| = 32$ was used, which was found sufficient for modeling the chosen targets according to a hyperparameter search conducted during an early experimental phase [40]. The model was implemented in Python using the PyTorch [41] ML framework.

3 Modeling targets

To study the research problem, two nonlinear circuits with short-term memory were chosen: the Pultec EQP-1A, a saturating analog equalizer (EQ) [42], and the ProCo RAT, an analog distortion effect [43]. The choice of the devices was guided by the circuits' nonlinear nature, which makes the modeling inherently harder and justifies the usage of NNs, as well as their varying number of user controls $|\boldsymbol{\phi}|$. The varying $|\boldsymbol{\phi}|$ was thought to influence the problem setting in a meaningful way by altering the dimensionality the model has to adhere to. In the following, both of the chosen circuits are given a brief technical description.

3.1 Pultec EQ

The Pultec EQ is a famous early studio program equalizer from the 1950s, originally manufactured by Pulse Techniques, Inc. [42]. It was one of the first EQs that allowed for continuous adjustment of multiple frequency bands. The sound of the Pultec is still revered due to its musical equalizing curves, as well as the warmth brought by the various transformers and vacuum tubes utilized in the circuit, which can be seen in the number of software emulations and hardware derivatives available in the market [44–46].

The circuit schematic of the Pultec EQ with the power conditioning circuitry removed is shown in Fig. 2. The system consists of a passive EQ stage and an active recovery/amplification stage, highlighted with colored boxes in Fig. 2. The EQ stage contains a low-shelving filter, a bell-shaped filter centered in the mid-to-high frequencies, and a high-shelving filter. The processing stages and the input and output of the circuit are connected using transformers, which produce harmonic distortion components, especially in the low frequencies [47]. Since the equalizer stage is passive, the signal level is attenuated as it passes through, which is then compensated for at the recovery stage. The recovery stage uses a balanced push-pull arrangement of vacuum tubes for the amplification, producing additional harmonic distortion components in the processed signal [48].

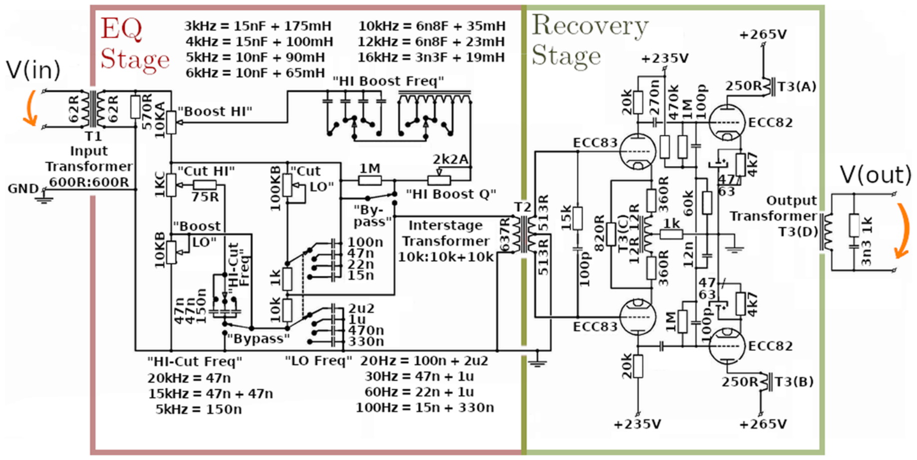

Fig. 2 Pultec EQ circuit diagram showing the equalizer and recovery stages, adapted from [49]

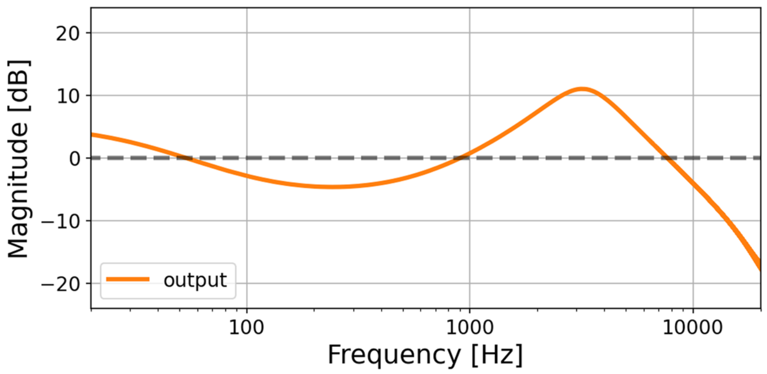

Fig. 3 Pultec EQ magnitude response with the low frequency boost and attenuation, middle frequency boost and resonance, and high frequency attenuation controls set to $\boldsymbol{\phi}_{\text{PULTEC}} = [g_{\text{low}}, a_{\text{low}}, g_{\text{mid}}, q_{\text{mid}}, a_{\text{high}}] = [1.0, 0.4, 0.75, 0.0, 0.75]$

The user controls of the circuit consist of both switchable and continuous controls. The switchable controls are the crossover frequency of the low-shelving filter f_{low}, the center frequency of the bell-shaped filter f_{mid}, and the crossover frequency of the high-shelving filter f_{high}. The continuous controls consist of, unusually, separate boosting g_{low} and attenuation a_{low} controls for the low-shelving filter, boosting g_{mid} and resonance q_{mid} controls for the bell-shaped filter, and an attenuation a_{high} control for the high-shelving filter. Since the controls are non-orthogonal and the effective boosting and attenuation crossover frequencies are not exactly the same, dialing in both boosting and attenuation simultaneously allows for acquiring more complicated frequency responses than would be immediately obvious.

For the scope of this work, the switchable characteristics of the Pultec EQ were set to constant values $(f_{\text{low}}, f_{\text{mid}}, f_{\text{high}}) = (20, 3\text{k}, 20\text{k})$ Hz, and only the continuously variable controls were used for conditioning the Pultec EQ-based models. Thus, the vector of conditioning values for the Pultec EQ becomes $\boldsymbol{\phi}_{\text{PULTEC}} = [g_{\text{low}}, a_{\text{low}}, g_{\text{mid}}, q_{\text{mid}}, a_{\text{high}}]$. An example magnitude response of the Pultec EQ is shown in Fig. 3, with the user controls set to $\boldsymbol{\phi}_{\text{PULTEC}} = [1.0, 0.4, 0.75, 0.0, 0.75]$, where the values from 0 to 1 represent the linear range of the potentiometer settings, from minimum to maximum.

3.2 ProCo RAT

The ProCo RAT is a popular analog distortion pedal originating from the late 1970s, originally manufactured by Pro Co Sound [43]. It is widely adopted by musicians of many disciplines to introduce richness and complexity to signals driven through it, produced by the highly nonlinear operation of the circuit.

The circuit schematic of the ProCo RAT with the power conditioning circuitry removed is shown in Fig. 4. A thorough analysis of the circuit is presented in [43]. The system consists of three stages: a clipping stage, a tone control stage and an output stage, highlighted with colored boxes in Fig. 4.

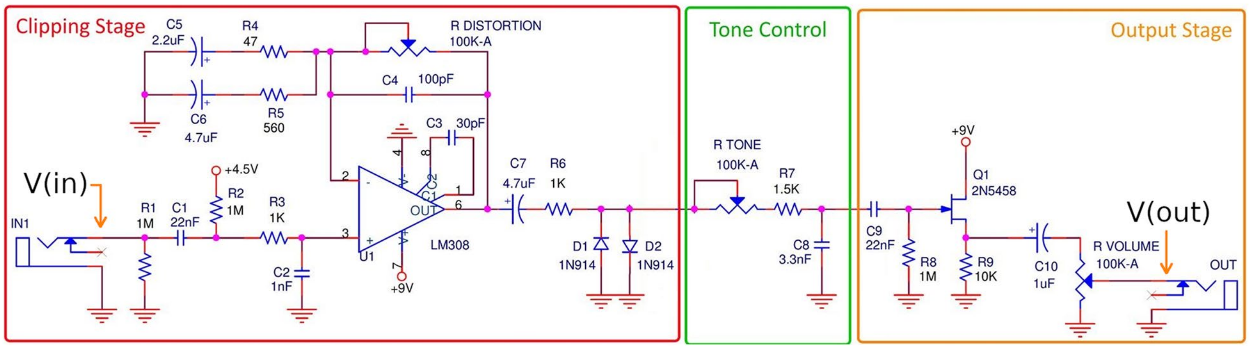

Fig. 4 ProCo RAT circuit diagram showing the clipping, tone control, and output stages, adapted from [43]

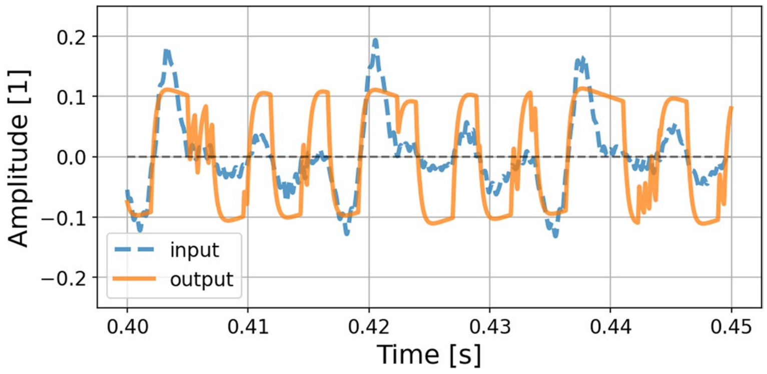

Fig. 5 ProCo RAT time-domain response with the distortion, tone, and output volume controls set to $\boldsymbol{\phi}_{\mathrm{RAT}} = [g_{\mathrm{dist}}, f_{\mathrm{tone}}, g_{\mathrm{vol}}] = [1.0, 1.0, 0.75]$

The clipping stage consists of an adjustable non-inverting amplifier driving a hard-clipping circuit, producing high amounts of distortion as the signal level is brought up. The non-inverting amplifier is implemented using an operational amplifier as the active element, and the hard-clipping is achieved with an anti-parallel connection of silicon diodes. The tone control stage consists of an adjustable first-order RC low-pass filter with a −6 -dB slope. The output stage consists of a source follower circuit driving an adjustable voltage divider, and is used to decouple the pedal electronics from the downstream circuitry. The source follower is implemented using a junction field effect transistor (JFET) as the active element, the inherent nonlinearities of which will also produce and add up to the harmonic distortion components produced by the circuit.

Each of the three processing stages contains a single continuous control. These controls are the distortion amount g_{dist} of the clipping stage (the gain of the non-inverting amplifier), the cutoff frequency of the tone stage f_{tone}, and the output volume g_{vol} (the setting of the voltage divider). All of these controls were used as conditioning for the ProCo RAT-based models, making the vector of conditioning values $\boldsymbol{\phi}_{\mathrm{RAT}} = [g_{\mathrm{dist}}, f_{\mathrm{tone}}, g_{\mathrm{vol}}]$. An example time-domain response of the ProCo RAT is shown in Fig. 5, with the user controls set to $\boldsymbol{\phi}_{\mathrm{RAT}} = [1.0, 1.0, 0.75]$, where again the values represent possible potentiometer settings, such that 1.0 corresponds to the maximum.

4 Data collection

To collect data for training, validation, and testing, SPICE simulations of the target circuits were utilized. The SPICE netlists were constructed using LTSpice [50], which were then invoked and controlled using the PyLTSpice [51] wrapper and library for Python. The various potentiometer laws encountered in the circuits were modeled according to the power approximations given in [52], and the user controls $\phi \in \boldsymbol{\phi}$ were parameterized so that their effective ranges were within the closed interval [0, 1]. The following subsections describe the source audio used for exciting the circuits, the parameter sampling procedure, and the datasets used for training the models.

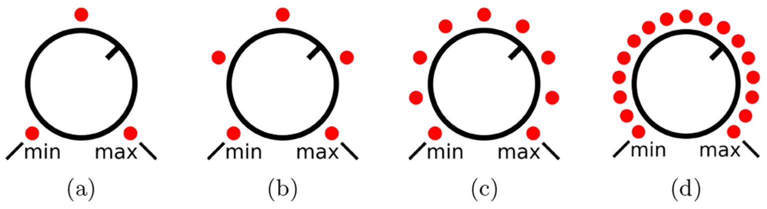

Fig. 6 Different parameter sampling densities δ used for the user controls. **a** $\delta = 3$. **b** $\delta = 5$. **c** $\delta = 9$. **d** $\delta = 17$

4.1 Source audio

The source audio used to excite the circuits consisted of 4 min of audio sampled at 44.1 kHz. The audio was divided into 1 min of guitar and 1 min of bass passages from [53] and [54], respectively, in addition to 1 min of synthesized logarithmic sine sweeps and 1 min of synthesized white noise passages, both at various amplitudes. This choice of source audio was inspired by successful experiments found in the literature [32, 34, 55], and the overall length was chosen to be similar to that which was used in related work [26, 33]. In order to ensure excitation of the circuits in their most nonlinear regions, all of the 1-min collections of recordings were normalized so that their maximum peak values reached −0.1 dBFS. Finally, to allow for using mini-batches later during training, the 4 min of source audio was split into 1-s segments, forming the partial dataset $\mathbb{D}' = \left\{\mathbf{u}^{(i)}\right\}_{i=1}^{240}$.

4.2 Parameter sampling

In order to construct datasets of the form given in Eq. (1), each input $\mathbf{u}^{(i)} \in \mathbb{D}'$ was driven through each of the targets, and for each simulation round, the user controls $\phi \in \boldsymbol{\phi}$ were sampled as:

$$\phi \sim \mathcal{U}\{0,1\}[\delta], \quad \delta \in \mathbb{Z}^+, \tag{6}$$

where $\mathcal{U}\{0,1\}$ is the discrete uniform distribution spanning [0, 1], δ is the sampling density or the number of sampled values within the interval, and $\mathbb{Z}^+$ is the set of positive integers. For example, when the sampling density is $\delta = 3$, each $\phi \in \boldsymbol{\phi}$ can take values from the set $\{0.0, 0.5, 1.0\}$, and each of these values has an equal likelihood of being selected.

For our experiments, the sampling densities $\delta = [3, 5, 9, 17]$ were considered, as illustrated in Fig. 6, on top of which a continuous sampling $\phi \sim \mathcal{U}\{0, 1\}$ was trialed, which we denote δ = "c" (*continuous*). The rationale in choosing the sampling densities was in halving the distances between the allowed sampling points at every increment of δ, such that for our choice, the distances became [0.5, 0.25, 0.125, 0.0625], or $\frac{1}{\delta-1}$, as well as the 64-bit floating point resolution for δ = "c".

4.3 Training datasets

To investigate the effects of the sampling density and the dataset size on the model generalization, pools of 50 datasets were constructed for each sampling density δ = [3,5,9,17,"c"] and target, denoted as:

$$\mathbb{D}_{\delta,j} = \left\{\left(\mathbf{u}^{(i)}, \mathbf{y}^{(i)}, \boldsymbol{\phi}^{(i)}\right)\right\}_{i=1}^{240}, \quad j \in [1, \ldots, 50]. \tag{7}$$

Due to the stochastic nature of the sampling procedure in Eq. (6), the contents of datasets $\mathbb{D}_{\delta,j}$ generated with the same sampling density δ are different and the number of possible device configurations $\boldsymbol{\phi}$ in $\mathbb{D}_{\delta,j}$ grows exponentially with the sampling density used. In contrast, since the sizes of the datasets $|\mathbb{D}_{\delta,j}| = 240$ are the same, the increasing diversity comes with the risk of $\boldsymbol{\phi} \in \mathbb{D}_{\delta,j}$ representing only a small subspace of the different possible configurations.

Noting the connection between the sampling density and the configuration diversity, the final datasets for training the models were constructed by creating stacks of $n = [1, 2, 4, 8, 16]$ datasets $\mathbb{D}_{\delta,j}$ for each sampling density δ. To gain robustness in the stacked datasets $\mathbb{D}_{\delta,j}$ possibly representing a particularly uneven distribution for the device configurations $\boldsymbol{\phi}$, five random draws from the $\binom{50}{n}$ possible subsets were created for each (δ, n) pair, with the corresponding datasets denoted as $\mathbb{D}_{\delta,n\times,k}$, $k \in [1, ..., 5]$. For example, when $(\delta, n) = (3, 4)$, the five datasets constructed for the configuration are denoted $\mathbb{D}_{3,4\times,k}$. With the considered choices for δ, n, and (#draws), the total number of datasets constructed for each target sums to 125.

5 Model training

To better understand the effect of dataset choice on model generalization, 125 models for each target device were trained, one for each distinct dataset $\mathbb{D}_{\delta,n\times,k}$ introduced in Sect. 4. The following subsections give a detailed description of the training procedure.

5.1 Validation and test sets

To create common benchmarks for evaluating the models trained using diverse training sets, separate validation and test sets were constructed for each target. The source audio for both the validation and test sets consisted of 30 s of unseen guitar and 30 s of unseen bass passages from the same distributions that were used during training, and the user controls $\phi \in \boldsymbol{\phi}$ were sampled continuously from $\mathcal{U}\{0,1\}$. These settings were meant to mimic the model operating conditions during inference.

To gain additional robustness to unlucky draws during the final testing phase, five iterations of the test sets were constructed for each target, over which the final results are further aggregated over, as explained in Sect. 6. As was done for the training datasets, the 30-s collections of recordings were normalized to maximum peak levels of -0.1 dBFS and split into 1-s segments to aid parallel processing. We denote the validation set $\mathbb{D}_{\text{val}}$ and the test sets $\mathbb{D}_{\text{test},l}$, $l \in [1, ..., 5]$.

5.2 Loss

To optimize the model weights $\boldsymbol{\theta}$, the error of the model prediction $\hat{\mathbf{y}}$ in comparison with the target output $\mathbf{y}$ is first evaluated using a chosen loss $\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})$ and the gradient of the loss w.r.t. the model weights $\nabla_{\boldsymbol{\theta}}\mathcal{L}$ is computed using the back-propagation algorithm [25]. In our work, the error-to-signal ratio (ESR) loss was used, defined as [30]:

$$\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) = \frac{\sum_{n=0}^{N-1} |y[n] - \hat{y}[n]|^2}{\sum_{n=0}^{N-1} |y[n]|^2}, \tag{8}$$

where $y[n]$ is the target output, $\hat{y}[n]$ the model prediction, N is the sequence length, and $|\cdot|$ is the absolute value operator. The term in the denominator normalizes the loss computations w.r.t. the energy of the target output in order to prevent high energy segments from dominating the weight optimization. For the scope of this work, no pre-emphasis filter was used before the loss computations, although it is known to be advantageous [56], since we wanted to test a basic NN model without extensions.

5.3 Training procedure

Instead of computing the gradient $\nabla_{\boldsymbol{\theta}}\mathcal{L}$ using the full training set $\mathbb{D}$, the gradient is approximated using mini-batches of examples $\mathbb{B} \sim \mathbb{D}$, and noisy estimates $\tilde{\nabla}_{\boldsymbol{\theta}}\mathcal{L}$ computed over $\mathbb{B}$ are used until each example $\left\{\left(\mathbf{u}^{(i)}, \mathbf{y}^{(i)}, \boldsymbol{\phi}^{(i)}\right)\right\} \in \mathbb{D}$ has been used [25]. This approach, commonly known as stochastic gradient descent (SGD), results in a higher number of optimizer calls for each training epoch in comparison with using the full gradient over $\mathbb{D}$. In our implementation, we use a relatively small batch size $|\mathbb{B}|$ of $2^5 = 32$ according to earlier experimental practice [26, 28].

When applying back-propagation to RNNs, the recursive computational graph resulting from evaluating Eq. (5) over the input sequence $\mathbf{u}$ is first unfolded to a regular directional computational graph, and the approximated gradients $\tilde{\nabla}_{\boldsymbol{\theta}}\mathcal{L}$ are computed using standard back-propagation rules [25]. This approach is commonly known as back-propagation through time (BPTT). Instead of unfolding the computational graph over the whole of $\mathbf{u}$, the input sequence is further split into shorter portions and the computational graphs are unfolded sequentially until the whole input sequence is traversed, calling the optimizer at the end of each portion [57]. This approach, commonly known as truncated BPTT, further increases the number of optimizer calls and speeds up training. In our implementation, we let the model state initialize for $N_{\text{INIT}} = 2^{10} = 1024$ samples before tracking the gradient, and the subsequent gradients were estimated using the same number of steps $N_{\text{TBPTT}} = 1024$ [14].

After computing $\tilde{\nabla}_{\boldsymbol{\theta}}\mathcal{L}$, the model weights $\boldsymbol{\theta}$ are stepped towards the negative gradient using some optimizer function. In our implementation, we used the Adam optimizer [58] with a learning rate γ of 1×10^{-3} and betas $(\beta_1, \beta_2) = (0.9, 0.999)$, which correspond to the default values for the method as implemented in PyTorch [41].

5.4 Normalizing compute

To ensure a fair comparison of models trained on datasets $\mathbb{D}$ of varying sizes, the number of optimizer steps the models were trained for was kept constant. The number of optimizer steps resulting from an epoch of training using truncated BPTT can be computed as:

$$s = N_B \left\lceil \frac{|\mathbf{u}| - N_{\text{INIT}}}{N_{\text{TBPTT}}} \right\rceil, \tag{9}$$

where $N_B = \left\lceil \frac{|\mathbb{D}|}{|\mathbb{B}|} \right\rceil$ is the number of mini-batches in an epoch, $|\mathbf{u}|$ is the input length in samples, and $\lceil\cdot\rceil$ is the ceiling function. The total number of optimizer steps s_{total} is then:

$$s_{\text{total}} = N_E s, \tag{10}$$

where N_E is the number of epochs.

In our experiments, all of the models were trained for $s_{\text{total}} = 430,000$ optimizer steps, or 10,000 batches of size $|\mathbb{B}| = 32$, after which the training was stopped. The models were trained on modern GPUs, and a typical training round took approximately 3–5 h to finish. During the progress of training, the validation loss over the validation sets $\mathbb{D}_{\text{val}}$ was seen to saturate for both targets, exemplified in Fig. 7 with 2 randomly chosen models. After the training was stopped, the model weights that produced the lowest validation loss were used for final evaluation.

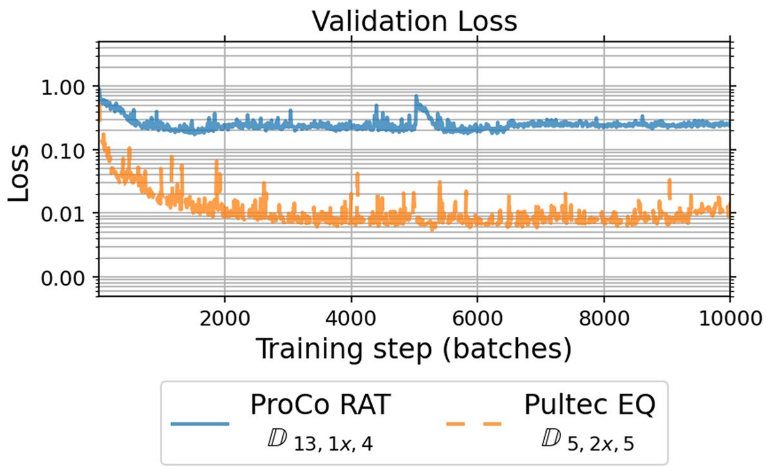

Fig. 7 Typical evolution of validation loss during training for both target devices

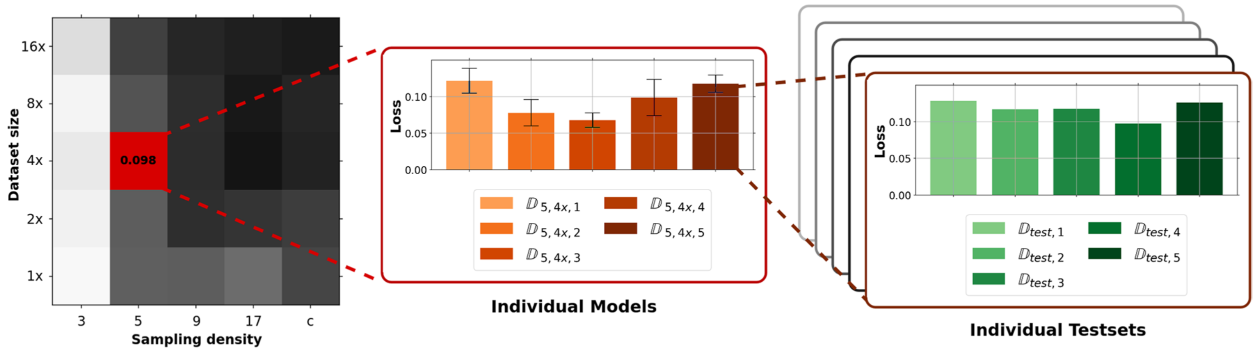

Fig. 8 Aggregation of the error metrics for each dataset configuration in the final results

6 Objective evaluation

To compare the performance of the models trained using the various training sets $\mathbb{D}_{k,n\times,k}$, the median ESR loss aggregated over the test sets $\mathbb{D}_{\text{test},l}$ is computed for each distinct model. In order to rule out the stochasticity of the parameter sampling procedure, the results for the models belonging to the same configurations $(\delta = \delta', n = n')$ are further aggregated. This aggregation procedure is illustrated in Fig. 8 for an example configuration $(\delta = 5, n = 4)$, where the ProCo RAT is the modeling target. As shown in the figure, the objective metrics used to assess any particular configuration is the aggregate over (#models) $\times$ (#test sets) $= 5 \times 5 = 25$ losses.

6.1 Objective results

The aggregated medians (η) of the ESR losses for the ProCo RAT models are visualized in Fig. 9. The best performing configurations are highlighted by using a white font, while configurations within a tolerance of 0.01 from the best are highlighted with a light gray font.

As can be seen from the figure, the models trained using the sampling density $\delta = 3$ clearly perform worse than the models trained using higher sampling densities. Generally speaking, increasing both the sampling density and the dataset size has a positive effect on the model performance, although the benefits seem to saturate along both axes and the contours are not strictly monotonic. Increasing the dataset size has the most positive effect on the model performance when using denser sampling grids for the device parameters, to the extent that the best performing models were trained using the densest sampling grids and larger dataset sizes $(\delta \geq 17, n \geq 4)$. Beyond the dataset configuration of $(\delta = 17, n = 4)$, no further increase in the model performance is achieved. Comparing the error metrics of the best performing models to those of related models in existing literature shows agreement in their magnitudes [31, 59], further validating the results.

The aggregated medians (η) of the ESR losses for the Pultec EQ models are visualized in Fig. 10. Note that the minimum of the z-axis is an order magnitude smaller in

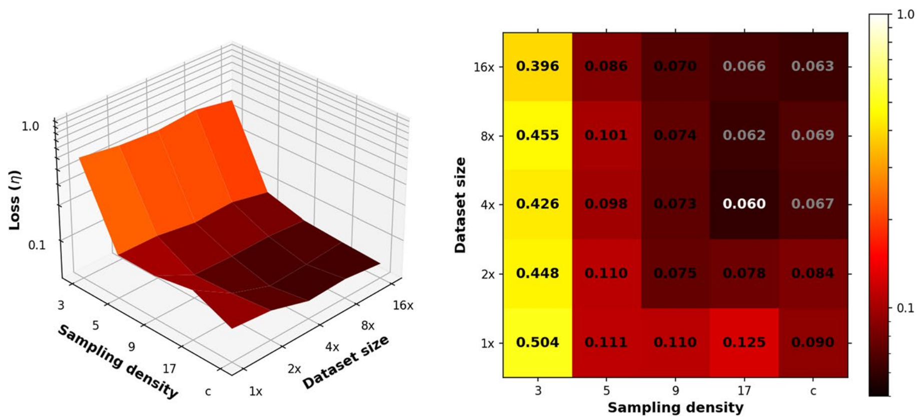

Fig. 9 The aggregated median (η) losses for the ProCo RAT models in the (left) 3-D plot and (right) matrix form

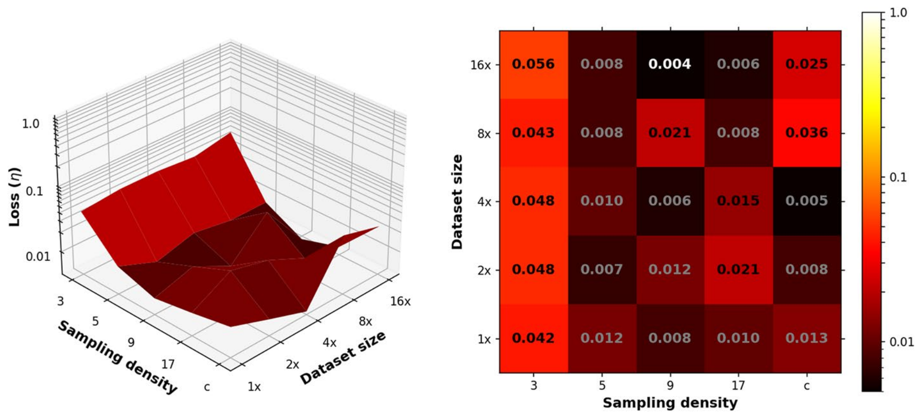

Fig. 10 The aggregated median (η) losses for the Pultec EQ models

comparison with earlier, due to the losses being much lower overall for the Pultec EQ models. We hypothesize that while the Pultec EQ has a larger number of user controls that the model has to adhere to in comparison with the ProCo RAT, it remains an easier target to model since it exhibits a greater degree of linearity. Again, the best performing configurations, and those within a tolerance of 0.01 from the best, are highlighted using white and light gray fonts, respectively.

Looking at Fig. 10 and similarly as before, the models trained using the sparsest sampling grid $\delta = 3$ perform worse than the rest. Beyond this sparsest sampling grid, the loss surface becomes noisy in its shape, and the models within a tolerance of 0.01 from the best configuration are scattered across the (δ, n) search space. Judging by the noisy shape of the loss surface and the small overall error metrics around this region, the results suggest that the models beyond the sparsest sampling grid of $\delta = 3$ have all reached convergence. Acknowledging the uncertainties brought by the noisiness, there is evidence that the models trained using the densest and largest configurations ($\delta =$ "c", $n \geq 8$) have slightly degraded performance, indicating that continuous sampling is a suboptimal choice. In light of these findings, we conclude that a sampling density of $\delta = 5$ adequately captures the device behavior for this particular target.

7 Subjective evaluation

In order to gain insight into how the acquired ESR losses relate to the perceptual quality of the trained models, an additional listening test was conducted. The listening test setup was similar to a MUSHRA test (ITU-R BS.1534) [60] and was conducted using the webMUSHRA [61] framework.

7.1 Listening test setup

A single model for each sampling density was chosen for the listening test. The models were picked such that for each sampling density, the best performing configuration w.r.t. the dataset size was chosen. From the five possible

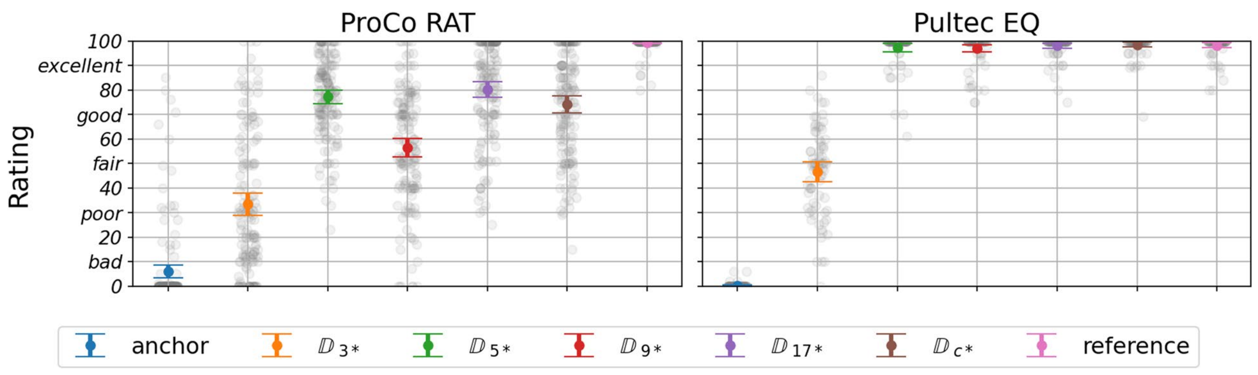

Fig. 11 Listening test results showing both the means and 95% confidence intervals of the ratings for both targets, indicating a monotonic increase in perceptual quality until a sampling density of $\delta = 5$, beyond which no further improvement is observed. The asterisk ($*$) denotes the best performing model for each sampling density

candidates $k \in [1, ..., 5]$, the median performing model was chosen as the representative one.

In order to make the listening test conditions more realistic, a new pool of audio material was collected, comprising short segments of music representing varying genres. The segments in the pool were processed using both target devices and the chosen models, randomizing the device parameters at the start of each segment. The segments processed by the target devices were used as the reference conditions. To create a low-quality anchor, the segments were additionally processed using a hyperbolic tangent function with $25\times$ of input gain and filtering the resulting outputs using a high-shelving filter with -18 dB of gain at Nyquist and the corner frequency set at 5.5 kHz. The final segments for the listening test were chosen by computing the segment-wise losses from the predictions generated by each of the chosen models, and picking a set of segments that produced an even distribution of low, mean, and high average losses, in order to have a fair choice of audio for the test. Finally, each segment was normalized to -23 dB LUFS, using the pyloudnorm library [62].

The listening test was conducted in sound-isolated listening booths at the Aalto Acoustics Lab using pairs of Sennheiser HD650 headphones. Fifteen experienced listeners without reported hearing impairments conducted the test, and no subjects were excluded during the post-hoc analysis of the ratings.

7.2 Subjective results

The results of the listening test are shown in Fig. 11. The asterisk ($*$) is used to denote the best performing model for each sampling density, according to the selection strategy underlined earlier. Similar to what was found in the objective evaluation, the models that were trained using the sparsest sampling grid $\delta = 3$ are clearly performing worse than the rest. From $\delta \geq 5$ onwards, the performance of the models is seen to saturate for both targets, although in the case of the ProCo RAT, the model representing the choice $\delta = 9$ is seen to perform worse than would be expected. In the case of the ProCo RAT, the saturation of the performance happens at a perceptual quality between *good* and *excellent*, while for the Pultec EQ, all of the models $\delta \geq 5$ are perceptually indistinguishable from the reference. While acquiring a model with excellent perceptual quality would have been desirable also for the ProCo RAT, we note the agreement between the acquired quality and the existing state-of-the-art for related devices [59] and hypothesize that given the highly nonlinear behavior of the device, reaching this level would have required further considerations such as perceptual weighting of the loss [56] or model anti-aliasing [63].

Listening to the segments processed by the $\delta = 9$ model for the ProCo RAT confirms that the perceptual quality of the model is noticeably worse than the others. To investigate this, we compute the short-time Fourier transform (STFT) loss [64] as well as the ESR loss over the listening test segments for the chosen models, shown in Fig. 12. While the ESR loss on the left of the figure shows the expected monotonic improvement of the loss metrics as the sampling density is increased, the STFT loss on the right clearly shows how the $\delta = 9$ model does not conform to the pattern. This finding suggests that while time-domain losses such as the ESR have been shown to be valid choices for training models of perceptually excellent quality [26, 27, 33], a frequency-domain loss can help in covering some aspects of the modeling problem not caught by focusing on the time-domain only.

Based on the patterns seen in the loss surface for the ProCo RAT models in Fig. 9, it would have been expected for the models trained using higher sampling densities to

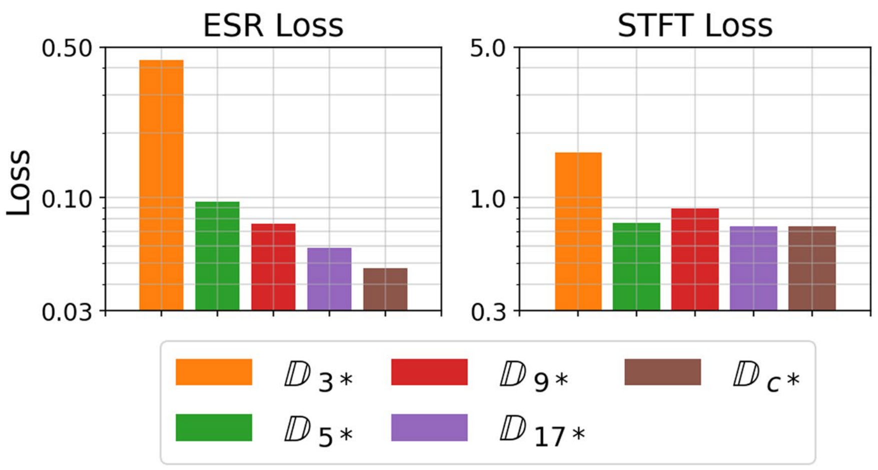

Fig. 12 ESR and STFT losses over the listening test segments for the chosen ProCo RAT models. The STFT loss demonstrates both the halt of perceptual improvement after a sampling density of $\delta = 5$ and the slightly degraded performance of model $\mathbb{D}_{9*}$ compared to the neighboring models

outperform the models trained using a sparse sampling grid of $\delta = 5$. However, the results of the listening tests show that, beyond a sampling density $\delta \geq 5$, no improvement in the model performance is achieved. This finding can be interpreted as showing, as was found by analyzing the behavior of the $\delta = 9$ model for the ProCo RAT, that the ESR is not a conclusive perceptual metric, and it should not be understood as a direct indicator of the model performance. Reminding ourselves of the error surfaces shown in Figs. 9 and 10, and keeping in mind the saturation of the perceptual quality, we find agreement in the overall trend of the results for both of the considered targets. In light of these findings, we conclude that a sampling grid $\delta = 5$ is sufficient for capturing the nonlinear behavior of the types of systems considered in this study, for the application of neural VA modeling.

8 Conclusions

This paper studied neural VA modeling of nonlinear parametric circuits, focusing on how the diversity in exposure to varying settings of the device user controls during training affects the network generalization. The problem was studied by generating a large corpus of training datasets for two chosen modeling targets using automated SPICE simulations, and training a proven RNN model for each of the datasets. The dataset properties that were altered during the generation were the sampling resolution of the device user controls, as well as the dataset size.

Our results demonstrate that a sampling density of five for the user controls is sufficient for modeling the types of devices considered in this work, i.e., nonlinear circuits with short-term memory and up to five user controls. This result is helpful when collecting training data for other similar devices, since generally no automatic way of setting the device parameters on an arbitrary grid exists and a sparse sampling of the parameter space is practically desirable, while collecting larger amounts of data using sparser grids only incurs a small additional cost.

In the future, the scope of the study could be extended to include, for example, multiple model architectures beyond the choice of RNNs, alternative loss functions, especially in the time-frequency domain, and other device types beyond nonlinear circuits with short-term memory. Further work is also needed to establish an explanation of why the densest possible sampling of the parameter space is not always the best choice for the considered task. The findings of this study can help reduce time and effort in collecting training data for deep NN models of audio devices.

Abbreviations

BPTT	Back-propagation through time
DL	Deep learning
ESR	Error-to-signal ratio
EQ	Equalizer
FC	Fully connected
GRU	Gated recurrent unit
JFET	Junction field effect transistor
ML	Machine learning
SGD	Stochastic gradient descent
STFT	Short-time Fourier transform
VA	Virtual analog

Acknowledgements

The work was completed whilst all authors were with the Aalto Acoustics Lab, Espoo, Finland. The authors acknowledge Aalto Science IT for the computational resources.

Authors' contributions

The study was conceptualized jointly by the authors. O.M. wrote the code base, produced the datasets, trained the models, and ran the experiments. The decisions for refining the methodology and the direction of the study during the experimental phase were made jointly by the authors. Also the manuscript was prepared, edited, and revised jointly. All authors read and approved the final manuscript.

Funding
This work was supported in part by the Nordic Sound and Music Computing Network—NordicSMC (NordForsk project number 86892).

Availability of data and materials
The datasets generated and analyzed during the current study will be made available on the accompanying website, http://research.spa.aalto.fi/publications/papers/jasm24-neural.

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 5 June 2023 Accepted: 22 April 2024
Published online: 20 May 2024

References
1. V. Välimäki, F. Fontana, J.O. Smith, U. Zolzer, Introduction to the special issue on virtual analog audio effects and musical instruments. IEEE Trans. Audio Speech Lang. Process. **18**(4), 713–714 (2010). https://doi.org/10.1109/TASL.2010.2046449
2. J. Pakarinen, V. Välimäki, F. Fontana, V. Lazzarini, J.S. Abel, Recent advances in real-time musical effects, synthesis and virtual analog models. EURASIP J. Adv. Signal Process. **2011**(1), 940784 (2011). https://doi.org/10.1155/2011/940784
3. J. Pakarinen, D.T. Yeh, A review of digital techniques for modeling vacuum-tube guitar amplifiers. Comput. Music J. **33**(2), 85–100 (2009). https://doi.org/10.1162/comj.2009.33.2.85
4. T. Vanhatalo, P. Legrand, M. Desainte-Catherine, P. Hanna, A. Brusco, G. Pille, Y. Bayle, A review of neural network-based emulation of guitar amplifiers. Appl. Sci. **12**(12), 5894 (2022). https://doi.org/10.3390/app12125894
5. O. Massi, A.I. Mezza, R. Giampiccolo, A. Bernardini, Deep learning-based wave digital modeling of rate-dependent hysteretic nonlinearities for virtual analog applications. EURASIP J. Audio Speech Music Process. **2023**(1) (2023). https://doi.org/10.1186/s13636-023-00277-8
6. J. Pekonen, V. Lazzarini, J. Timoney, J. Kleimola, V. Välimäki, Discrete-time modelling of the Moog sawtooth oscillator waveform. EURASIP J. Adv. Signal Process. **2011**(1), 785103 (2011). https://doi.org/10.1155/2011/785103
7. L. Gabrielli, S. D'Angelo, L. Turchet, in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*. Analysis and emulation of early digitally-controlled oscillators based on the Walsh-Hadamard transform (Birmingham City University, Birmingham, 2019), pp. 319–325
8. A. Huovilainen, in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*. Non-linear digital implementation of the Moog ladder filter (Federico II University of Naples, Naples, 2004), pp. 61–64
9. M. Rest, J.D. Parker, K.J. Werner, in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*. WDF modeling of a Korg MS-50 based non-linear diode bridge VCF (University of Edinburgh, Edinburgh, 2017), pp. 145–151
10. V. Lazzarini, J. Timoney, Improving the Chamberlin digital state variable filter. J. Audio Eng. Soc. **70**(6), 446–456 (2022). https://doi.org/10.17743/jaes.2022.0001
11. O. Kröning, K. Dempwolf, U. Zölzer, in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*. Analysis and simulation of an analog guitar compressor (IRCAM, Paris, 2011), pp. 205–208
12. A. Wright, V. Välimäki, in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*. Grey-box modelling of dynamic range compression (The University of Music and Performing Arts, Vienna, 2022), pp. 304–311
13. K.J. Werner, W.R. Dunkel, G. Germain, in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*. A computational model of the Hammond organ vibrato/chorus using wave digital filters (Brno University of Technology, Brno, 2016), pp. 271–277
14. A. Wright, V. Välimäki, Neural modeling of phaser and flanging effects. J. Audio Eng. Soc. **69**(7), 517–529 (2021). https://doi.org/10.17743/jaes.2021.0029
15. D.T. Yeh, Digital implementation of musical distortion circuits by analysis and simulation. Ph.D. thesis, Stanford University, Stanford, US (2009)
16. D.T. Yeh, J.O. Smith, in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*. Simulating guitar distortion circuits using wave digital and nonlinear state-space formulations (Helsinki University of Technology, Espoo, 2008), pp. 19–26
17. K.J. Werner, Virtual analog modeling of audio circuitry using wave digital filters. Ph.D. thesis, Stanford University, Stanford, CA (2016)
18. F. Eichas, U. Zölzer, in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*. Black-box modeling of distortion circuits with block-oriented models (Brno University of Technology, Brno, 2016), pp. 39–46
19. T. Helie, Volterra series and state transformation for real-time simulations of audio circuits including saturations: application to the Moog ladder filter. IEEE Trans. Audio Speech Lang. Process. **18**(4), 747–759 (2010). https://doi.org/10.1109/TASL.2009.2035211
20. M.J. Kemp, in *106th Audio Engineering Society Convention*. Analysis and simulation of non-linear audio processes using finite impulse responses derived at multiple impulse amplitudes (Audio Engineering Society, Munich, 1999)
21. R. Kiiski, F. Esqueda, V. Välimäki, in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*. Time-variant gray-box modeling of a phaser pedal (Brno University of Technology, Brno, 2016), pp. 31–38
22. C. Darabundit, R. Wedelich, P. Bischoff, in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*. Digital grey box model of the Uni-Vibe effects pedal (Birmingham City University, Birmingham, 2019), pp. 261–268
23. A. Krizhevsky, I. Sutskever, G.E. Hinton, in *Advances in Neural Information Processing Systems*. ImageNet classification with deep convolutional neural networks, vol. 25 (Curran Associates Inc., Lake Tahoe, 2012), pp. 1106–1114
24. G. Hinton, L. Deng, D. Yu, G.E. Dahl, A.R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T.N. Sainath, B. Kingsbury, Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. IEEE Signal Process. Mag. **29**(6), 82–97 (2012). https://doi.org/10.1109/MSP.2012.2205597
25. I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning*. Adaptive computation and machine learning (the MIT Press, Cambridge, 2016)
26. A. Wright, E.P. Damskägg, L. Juvela, V. Välimäki, Real-time guitar amplifier emulation with deep learning. Appl. Sci. **10**(3), 766 (2020). https://doi.org/10.3390/app10030766
27. M.A. Martínez Ramírez, E. Benetos, J.D. Reiss, Deep learning for black-box modeling of audio effects. Appl. Sci. **10**(2), 638 (2020). https://doi.org/10.3390/app10020638
28. C.J. Steinmetz, J.D. Reiss, in *152nd Audio Engineering Society Convention*. Efficient neural networks for real-time modeling of analog dynamic range compression (Audio Engineering Society, The Hague, 2022)
29. T. Schmitz, J.J. Embrechts, in *144th Audio Engineering Society Convention*. Nonlinear real-time emulation of a tube amplifier with a long short term memory neural-network (Audio Engineering Society, Milan, 2018)
30. E.P. Damskägg, L. Juvela, E. Thuillier, V. Välimäki, in *Proceedings of the International Conference on Acoustics. Speech and Signal Processing (ICASSP)*, Deep learning for tube amplifier emulation (IEEE, Brighton, 2019), pp. 471–475. https://doi.org/10.1109/ICASSP.2019.8682805
31. A. Wright, E.P. Damskägg, V. Välimäki, in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*. Real-time black-box modelling with recurrent neural networks (Birmingham City University, Birmingham, 2019), pp. 173–180
32. J.D. Parker, F. Esqueda, A. Bergner, in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*. Modelling of nonlinear state-space systems using a deep neural network (Birmingham City University, Birmingham, 2019), pp. 165–172
33. A. Peussa, E.P. Damskägg, T. Sherson, S.I. Mimilakis, L. Juvela, A. Gotsopoulos, V. Välimäki, in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*. Exposure bias and state matching in recurrent neural network virtual analog models (The University of Music and Performing Arts, Vienna, 2021), pp. 284–291

34. F. Esqueda, B. Kuznetsov, J.D. Parker, in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*. Differentiable white-box virtual analog modeling (The University of Music and Performing Arts, Vienna, 2021), pp. 41–48
35. S. Hawley, B. Colburn, S.I. Mimilakis, in *147th Audio Engineering Society Convention*. Profiling audio compressors with deep neural networks (Audio Engineering Society, New York, 2019)
36. S. Nercessian, A. Sarroff, K.J. Werner, in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Lightweight and interpretable neural modeling of an audio distortion effect using hyperconditioned differentiable biquads (IEEE, Toronto, 2021), pp. 890–894. https://doi.org/10.1109/ICASSP39728.2021.9413996
37. L. Juvela, E.P. Damskägg, A. Peussa, J. Mäkinen, T. Sherson, S.I. Mimilakis, K. Rauhanen, A. Gotsopoulos, in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. End-to-end amp modeling: from data to controllable guitar amplifier models (Rhodes Island, 2023). https://doi.org/10.1109/ICASSP49357.2023.10094769
38. E.R. Scheinerman, *Invitation to dynamical systems* (Prentice Hall, Upper Saddle River, 1996)
39. K. Cho, B. van Merrienboer, D. Bahdanau, Y. Bengio, in *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. On the properties of neural machine translation: encoder-decoder approaches (Association for Computational Linguistics, Doha, 2014). https://doi.org/10.48550/arXiv.1409.1259
40. O. Mikkonen, Learning parameter spaces in neural modeling of audio circuits. Master's thesis, Aalto University, Espoo, Finland (2022)
41. A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, in *33rd Conference on Neural Information Processing Systems (NeurIPS)*. PyTorch: an imperative style, high-performance deep learning library, vol. 32 (Curran Associates Inc., Vancouver, 2019), pp. 8024–8035
42. H. Robjohns, Pulse techniques EQP-1A. Sound on Sound **34**(4), 114-118 (2019)
43. Electrosmash. ProCo RAT analysis. https://www.electrosmash.com/proco-rat. Accessed 17 June 2022
44. Universal Audio. Pultec passive EQ collection. https://www.uaudio.com/uad-plugins/equalizers/pultec-passive-eq-collection.html. Accessed 02 Nov 2022
45. Warm Audio. EQP-WA Pultec-style tube equalizer. https://warmaudio.com/eqp-wa/. Accessed 02 Nov 2022
46. TUBE-TECH. PE 1C program equalizer. http://www.tube-tech.com/pe-1c-program-equalizer/. Accessed 02 Nov 2022
47. C.D.R. de Paiva, J. Pakarinen, V. Välimäki, M. Tikander, Real-time audio transformer emulation for virtual tube amplifiers. EURASIP J. Adv. Signal Process. **2011**(1), 347645 (2011). https://doi.org/10.1155/2011/347645
48. E. Barbour, The cool sound of tubes. IEEE Spectr. **35**(8), 24–35 (1998). https://doi.org/10.1109/6.708439
49. Gyraf Audio. Do-A-Pultec page. https://www.gyraf.dk/gy_pd/pultec/pultec.htm. Accessed 07 Jan 2022
50. Analog Devices. LTspice simulator. https://www.analog.com/en/design-center/design-tools-and-calculators/ltspice-simulator.html. Accessed 17 June 2022
51. N. Brum. PyLTSpice. https://github.com/nunobrum/PyLTSpice. Accessed 19 May 2022
52. B. Holmes, M. van Walstijn, in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*. Potentiometer law modelling and identification for application in physics-based virtual analogue circuits (Birmingham City University, Birmingham, 2019), pp. 332–339
53. C. Kehling, J. Abeßer, C. Dittmar, G. Schuller, in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*. Automatic tablature transcription of electric guitar recordings by estimation of score- and instrument-related parameters (Fraunhofer IIS and Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, 2014), pp. 219–226
54. J. Abeßer, P. Kramer, C. Dittmar, G. Schuller, in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*. Parametric audio coding of bass guitar recordings using a tuned physical modeling algorithm (Maynooth University, Maynooth, 2013), pp. 154–161
55. B. Kuznetsov, J.D. Parker, F. Esqueda, in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*. Differentiable IIR filters for machine learning applications (The University of Music and Performing Arts, Vienna, 2020), pp. 297–303
56. A. Wright, V. Välimäki, in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Perceptual loss function for neural modeling of audio systems (IEEE, Barcelona, 2020), pp. 251–255. https://doi.org/10.1109/ICASSP40776.2020.9052944
57. J.L. Elman, Finding structure in time. Cogn. Sci. **14**(2), 179–211 (1990). https://doi.org/10.1207/s15516709cog1402_1
58. D.P. Kingma, J. Ba, in *International Conference on Learning Representations*. Adam: a method for stochastic optimization (San Diego, 2015)
59. D. Südholt, A. Wright, C. Erkut, V. Välimäki, Pruning deep neural network models of guitar distortion effects. IEEE Trans. Audio Speech Lang. Process. **31**, 256–264 (2023). https://doi.org/10.1109/TASLP.2022.3223257
60. International Telecommunication Union, BS.1534: method for the subjective assessment of intermediate quality level of audio systems. Recommendation BS.1534. (2015). https://www.itu.int/rec/R-REC-BS.1534/en. Accessed 08 June 2022
61. M. Schoeffler, S. Bartoschek, F.R. Stöter, M. Roess, S. Westphal, B. Edler, J. Herre, webMUSHRA—a comprehensive framework for web-based listening tests. J. Open Res. Softw. **6**(1) (2018). https://doi.org/10.5334/jors.187
62. C.J. Steinmetz, J.D. Reiss, in *150th Audio Engineering Society Convention*, Pyloudnorm: a simple yet flexible loudness meter in Python (Audio Engineering Society, Online, 2021)
63. T. Vanhatalo, P. Legrand, M. Desainte-Catherine, P. Hanna, G. Pille, Evaluation of real-time aliasing reduction methods in neural networks for nonlinear audio effects modelling. J. Audio Eng. Soc. **72**(3), 114–122 (2024). https://doi.org/10.17743/jaes.2022.0122
64. C.J. Steinmetz, J.D. Reiss, in *Digital Music Research Network One-day Workshop*. Auraloss: audio-focused loss functions in PyTorch (Queen Mary University of London, London, 2020)