EXPOSURE BIAS AND STATE MATCHING IN RECURRENT NEURAL NETWORK VIRTUAL ANALOG MODELS

Aleksi Peussa, Eero-Pekka Damskägg, Thomas Sherson, Stylianos I. Mimilakis, Lauri Juvela, Athanasios Gotsopoulos

Neural DSP Technologies
Helsinki, Finland
{aleksi, eero-pekka, thom, stelios, lauri, tan}@neuraldsp.com

Vesa Välimäki

Acoustics Lab, Dept. of Signal Processing and Acoustics
Aalto University
Espoo, Finland
vesa.valimaki@aalto.fi

ABSTRACT

Virtual analog (VA) modeling using neural networks (NNs) has great potential for rapidly producing high-fidelity models. Recurrent neural networks (RNNs) are especially appealing for VA due to their connection with discrete nodal analysis. Furthermore, VA models based on NNs can be trained efficiently by directly exposing them to the circuit states in a gray-box fashion. However, exposure to ground truth information during training can leave the models susceptible to error accumulation in a free-running mode, also known as "exposure bias" in machine learning literature. This paper presents a unified framework for treating the previously proposed state trajectory network (STN) and gated recurrent unit (GRU) networks as special cases of discrete nodal analysis. We propose a novel circuit state-matching mechanism for the GRU and experimentally compare the previously mentioned networks for their performance in state matching, during training, and in exposure bias, during inference. Experimental results from modeling a diode clipper show that all the tested models exhibit some exposure bias, which can be mitigated by truncated backpropagation through time. Furthermore, the proposed state matching mechanism improves the GRU modeling performance of an overdrive pedal and a phaser pedal, especially in the presence of external modulation, apparent in a phaser circuit.

1. INTRODUCTION

VA modeling [1] attempts to reproduce the desirable sonic characteristics of analog audio processing devices while side-stepping their inconveniences by operating in a purely digital domain. While white-box VA modeling approaches [2, 3] can produce accurate digital models, the creation of these models requires expert knowledge in circuit analysis and digital signal processing. Furthermore, the process is often labor intensive. This creates a clear appeal in automating VA modeling via black-box methods for rapid development and deployment.

Various black-box system identification methods for guitar amplifiers and distortion effects have been proposed [4, 5], but the methodology is constrained by the coupling between the parameter estimation method and a Wiener-Hammerstein model topology. Meanwhile, NN and generally deep learning methods for VA have gained popularity not only due to their black-box convenience, but also their flexible model architecture choice. Generic NN black-box models include a feedforward variant of the WaveNet [6, 7] and encoder-decoder models with recurrent bottleneck processors [8]. A deep learning framework further enables the inclusion (and end-to-end tuning) of digital signal processing (DSP) components, such as oscillators [9] and infinite impulse response filters [10, 11].

Typical audio processing circuits are stateful nonlinear systems, with relatively few reactive components. As such, comparisons arise naturally between classical circuit analysis techniques, such as the Discretized Kirchhoff nodal analysis (DK-method) [12, 13], and stateful NN models, such as recurrent neural networks (RNNs). RNNs with gated activations (for example, the long short-term memory (LSTM) [14] network and the gated recurrent unit (GRU) [15]) have been successfully applied in a black-box VA setting [16–18]. Although these RNNs are stateful, their state bears no direct relationship to the analog circuit state they aim to emulate. Meanwhile, the recently proposed State Trajectory Network (STN) [19] directly draws inspiration from the DK-method by tying the output of the NN model with the analog circuit state. The STN leverages the observed state information for efficient training, but its generalization to inference without available ground-truth states is not yet well understood. On the other hand, the present black-box NN models are relatively slow to train, which could be mitigated by utilizing available circuit states.

Recurrent (and recursive) networks are widely used for sequence modeling tasks in the nearby fields of natural language processing (NLP) and speech synthesis. Invariably, the models share a common discrepancy between training and inference time. That is, at training time, recursively computed states are often substituted with ground truth information. A common example of this is to replace a recursive model output with the true target output. However, at inference time such a substitution is not possible and the model must instead recursively rely on its own previous predictions (and states) as conditioning for its subsequent outputs. This discrepancy between training and test time conditions is known as "exposure bias". Training techniques that "expose" the model to ground truth information in this way are referred to as "teacher forcing", analogous to a teacher correcting a student's mistakes while sequentially working through a problem.

The exposure bias problem has attracted considerable research attention especially in NLP, with proposed mitigation techniques including scheduled sampling [20], decoding model outputs from multiple candidate trajectories [21], optimizing metrics at a sequence level [22], adversarial training schemes [23, 24], and reinforcement learning [25]. Though the exposure bias phenomenon is acknowledged in NLP, it remains difficult to quantify due to the natural random variation present in the data. In contrast, the elec-

trical circuits in VA modeling are practically deterministic, which facilitates studying exposure bias in a VA context. An initial investigation into state matching for GRUs and the effect of truncated backpropagation through time (TBPTT) [26] for various recurrent VA models was presented in [27].

In this paper, we expand upon previous work by exploring the impact of state matching and exposure bias on NN based VA models. In Sec. 2 we demonstrate how existing STN and GRU based networks can be interpreted as learnable DK-method-like algorithms and analyze the effect of teacher forcing on the training dynamics of such RNNs. In Sec. 3 we propose a GRU variant which incorporates a state matching capability similar to the STN via a state-based loss function. We then demonstrate a range of experimental results in Sec. 4, where we firstly study the exposure bias problem as a generalization gap between the quality of model outputs generated during training and during inference, for a simple diode clipper circuit. We also conduct further experiments to quantify the effects of model architecture choice and state matching during training on a more complex overdrive circuit and a modulated phaser. Finally in Sec. 5, we finish out the paper with our concluding remarks.

2. A GENERALIZED INTERPRETATION OF RECURSIVE BLACK-BOX METHODS

Nonlinear state space methods are a common tool in the world of VA modeling. Among such approaches is the so called DK-method [12, 13] which decomposes a discretized circuit model into a cascade of a linear transformation and a nonlinear mapping f, whose form is implied by the underlying circuit topology. Under the assumption that f is functional, the recursive update equations for such an approach are given by

$$\begin{aligned} \mathbf{x}_t &= \mathbf{A}\mathbf{x}_{t-1} + \mathbf{B}\mathbf{u}_t + \mathbf{C}\mathbf{w}_t,\ \mathbf{w}_t = f\left(\mathbf{x}_{t-1}, \mathbf{u}_t\right) \\ \mathbf{y}_t &= \mathbf{A}_E\mathbf{x}_t + \mathbf{B}_E\mathbf{u}_t + \mathbf{C}_E\mathbf{w}_t, \end{aligned} \tag{1}$$

where the vectors $\mathbf{x}_t \in \mathbb{R}^N$, $\mathbf{u}_t \in \mathbb{R}^M$, $\mathbf{w}_t \in \mathbb{R}^P$, and $\mathbf{y}_t \in \mathbb{R}^Q$ are the system states, inputs, nonlinear activations and outputs at time step t respectively and $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{A}_E$, $\mathbf{B}_E$, and $\mathbf{C}_E$ are a set of appropriately sized matrices. While seemingly distinct from the world of NN black-box modeling, in the following we demonstrate that both STNs and GRUs can be interpreted as learnable variants of such a DK-type method.

We firstly consider STNs. According to [19], one can freely choose the states of the circuit(s) to be modeled. We explicitly select the device *outputs* to be a subset of these states and assume that the output signals correspond to the last Q observed states. We do so, because during initial experimentation with STNs it was found that this configuration leads to the best performance in modeling audio circuits. More formally, we define the matrices

$$\begin{aligned} \mathbf{A} = \mathbf{I}_{N\times N},\ \mathbf{B} = \mathbf{0}_{N\times M},\ \mathbf{C} = \mathbf{I}_{N\times N} \\ \mathbf{A}_E = \begin{bmatrix}\mathbf{0}_{Q\times(N-Q)} & \mathbf{I}_{Q\times Q}\end{bmatrix},\ \mathbf{B}_E = \mathbf{0}_{Q\times M},\ \mathbf{C_E} = \mathbf{0}_{Q\times Q}, \end{aligned} \tag{2}$$

where $\mathbf{I}_{Q\times Q}$ and $\mathbf{0}_{Q\times M}$ denote a Q dimensional identity matrix and a Q by M zeros matrix respectively. By then defining the nonlinear function $f_s : \mathbb{R}^{N+M} \to \mathbb{R}^N$ to be a multi-layer perceptron (MLP), it follows that the STN update equations are equivalent to

$$\begin{aligned} \mathbf{x}_t &= \mathbf{x}_{t-1} + \mathbf{w}_t,\ \mathbf{w}_t = f_s\left(\mathbf{x}_{t-1}, \mathbf{u}_t\right) \\ \mathbf{y}_t &= \begin{bmatrix}\mathbf{0}_{Q\times(N-Q)} & \mathbf{I}_{Q\times Q}\end{bmatrix}\mathbf{x}_t, \end{aligned} \tag{3}$$

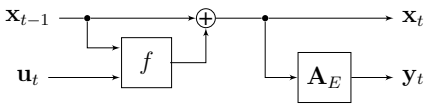

Figure 1: *Signal Flow Diagram for STN and GRU.*

and form a restricted subset of a learnable DK-method. Note that all of the learnable system parameters are incorporated into the nonlinear mapping, f_s, as weights and biases of the MLP.

The connection with GRUs is also straightforward to derive. First, consider the state update equations for a standard GRU [15],

$$\begin{aligned} \mathbf{x}_t &= \mathbf{z}_t \odot \mathbf{x}_{t-1} + (\mathbf{1} - \mathbf{z}_t) \odot \mathbf{g}_t \\ \mathbf{z}_t &= \sigma\left(\mathbf{W}_{x,z}\mathbf{x}_{t-1} + \mathbf{b}_{x,z} + \mathbf{W}_{u,z}\mathbf{u}_t + \mathbf{b}_{u,z}\right) \\ \mathbf{r}_t &= \sigma\left(\mathbf{W}_{x,r}\mathbf{x}_{t-1} + \mathbf{b}_{x,r} + \mathbf{W}_{u,r}\mathbf{u}_t + \mathbf{b}_{u,r}\right) \\ \mathbf{g}_t &= \tanh\left(\mathbf{r}_t \odot \left(\mathbf{W}_{x,g}\mathbf{x}_{t-1} + \mathbf{b}_{x,g}\right) + \mathbf{W}_{u,g}\mathbf{u}_t + \mathbf{b}_{u,g}\right) \\ \mathbf{y}_t &= \mathbf{W}_y\mathbf{x}_t + \mathbf{b}_y, \end{aligned} \tag{4}$$

where $\odot$ denotes the Hadamard product. By then introducing the additional vector variable $\mathbf{w}_n \in \mathbb{R}^N$ and nonlinear mapping $f_g : \mathbb{R}^{N+M} \to \mathbb{R}^N$ such that

$$\mathbf{w}_t = (\mathbf{1} - \mathbf{z}_t) \odot (\mathbf{g}_t - \mathbf{x}_{t-1}) = f_g\left(\mathbf{x}_{t-1}, \mathbf{u}_t\right), \tag{5}$$

and by defining the additional following matrices

$$\begin{aligned} \mathbf{A} = \mathbf{I}_{N\times N},\ \mathbf{B} = \mathbf{0}_{N\times M},\ \mathbf{C} = \mathbf{I}_{N\times N} \\ \mathbf{A}_E = \mathbf{W}_y,\ \mathbf{B}_E = \mathbf{0}_{Q\times M},\ \mathbf{C_E} = \mathbf{0}_{Q\times N}, \end{aligned} \tag{6}$$

the updates in (4) can be equivalently written as

$$\begin{aligned} \mathbf{x}_t &= \mathbf{x}_{t-1} + \mathbf{w}_t,\ \mathbf{w}_t = f_g\left(\mathbf{x}_{t-1}, \mathbf{u}_t\right) \\ \mathbf{y}_t &= \mathbf{W}_y\mathbf{x}_t + \mathbf{b}_y. \end{aligned} \tag{7}$$

Note that as in the case of the STNs, the GRU architecture has again imposed the restriction that $P = N$.

Comparing the update equations in (3) and (7), one can note the near identical nature of the recursive updates ($\mathbf{x}_t$ and $\mathbf{w}_t$) with the only distinction being the architecture of the nonlinear mapping adopted. Ignoring the additional additive bias term in (7) for the sake of brevity, it follows that the signal flow of both methods can be represented by the diagram in Fig. 1.

Due to the well known universal (memoryless) function approximation properties of MLPs [28], the difference in nonlinearity might suggest that the STN could be more flexible than its GRU counterpart. For the training routine proposed in [19] which involves ground truth state information, this may well be the case at training time. However, this approach can have a significant impact on inference performance when the ground truth states are no longer available. In contrast, the particular topology of f_g is well known to lend itself to more effective training via truncated back propagation through time [26] which we postulate can lead to better model performance at inference time.

2.1. STN, Teacher-Forcing, and Exposure Bias

Given the previously mentioned connection between STNs and recurrent mechanisms for black-box modeling, we also link the corresponding proposed training procedure with that of recurrent

mechanisms. To that aim, allow $\hat{\mathbf{x}} \in \mathbb{R}^N$ to be the approximation of the target state signal $\mathbf{x} \in \mathbb{R}^N$ computed using f_s from (3). From (3), $\mathbf{x}$ can be seen as the vector that contains both the output and the remaining state variables. For optimizing the parameters of f_s, stochastic gradient descent is applied using ground truth data obtained from the system to be modeled at *every* time-step t. It follows that an STN is essentially a recurrent mechanism optimized using the *teacher-forcing* technique, proposed in [29, 30]. To highlight the exposure bias in STNs, let us focus on the learning of f_s. For the sake of brevity, we will focus on the last layer of f_s, parameterized by the matrix $\mathbf{\Theta} \in \mathbb{R}^{N \times N_\text{h}}$ where N_h denotes the dimensionality of the latent vectors computed by the computational layers of f_s. It follows that the partial derivatives can be calculated[1] as

$$\frac{\partial \text{E}}{\partial \mathbf{\Theta}} = \frac{\partial \text{E}}{\partial \hat{\mathbf{x}}} \frac{\partial \hat{\mathbf{x}}}{\partial \mathbf{w}} \frac{\partial \mathbf{w}}{\partial \tilde{\mathbf{w}}} \frac{\partial \tilde{\mathbf{w}}}{\partial \mathbf{\Theta}}, \tag{8}$$

where E is the outcome of a differentiable loss function that compares the predicted and target outputs $\hat{\mathbf{x}}$ and $\mathbf{x}$. From (3), the output of f_s is denoted by $\mathbf{w} \in \mathbb{R}^N$. The tilde symbol " $\tilde{\bullet}$ " is used to denote the computed latent vector prior to the application of the element-wise non-linear activation function and thus, $\frac{\partial \mathbf{w}}{\partial \tilde{\mathbf{w}}}$ refers to the first derivative of the element-wise non-linear activation function(s) contained in f_s.

As can be seen in (8), the updates of the parameters of the f_s do not consider the temporal contribution of previous time-steps $t' < t$. Essentially, f_s does not model temporal dependencies longer than a single advance from $t-1$ to t, i.e., the shortest possible trajectory is considered during training. Consequently, f_s is exposed only to vectors drawn from the training data-set and not the predicted ones by evaluating f_s at some previous time-step. This could limit the generalization performance of the model during inference [25], because at inference time only the approximations $\mathbf{x}$ are available for conditioning. It is therefore useful to expose f_s to approximations of the input vectors, so the model may learn to recover from its own mistakes.

A straightforward way to expose f_s to the previously mentioned bias can be achieved by allowing f_s to freely-run for T time-steps during the training procedure. In the free-running mode, the output of f_s is used as input for the next time-step and the optimization of $\mathbf{\Theta}$ can be done using back-propagation through-time (BPTT) [26] and its time-truncated (TBPTT) variant [31]. With this in mind, we propose to consider the temporal contributions of previous time-steps t' to the loss at time-step t as

$$\frac{\partial \text{E}_t}{\partial \mathbf{\Theta}} = \sum_{t'=0}^{t} \frac{\partial \text{E}_t}{\partial \hat{\mathbf{x}}_t} \frac{\partial \hat{\mathbf{x}}_t}{\partial \mathbf{w}_t} \frac{\partial \mathbf{w}_t}{\partial \mathbf{w}_{t'}} \frac{\partial_\text{inst}\, \mathbf{w}_{t'}}{\partial \mathbf{\Theta}}, \tag{9}$$

where $\frac{\partial_\text{inst}\, \mathbf{w}_{t'}}{\partial \mathbf{\Theta}}$ is the instantaneous gradient. The instantaneous gradient computes the gradient of $\mathbf{w}$ with respect to $\mathbf{\Theta}$ only at t', disregarding other time-steps. In contrast to the instantaneous gradient, $\frac{\partial \mathbf{w}_t}{\partial \mathbf{w}_{t'}}$ is calculated by considering the dependencies from all the previously evaluated time-steps up to t including the applied element-wise non-linear function. It is important to notice that for $T = 1$, (9) boils down to (8), i.e., the training scheme of the STN as originally proposed in [19]. A caveat for letting $T > 1$, is that the Jacobian $\frac{\partial \mathbf{w}_t}{\partial \mathbf{w}_{t'}}$ can perturb exceedingly small or large values, due to the computation of a product that includes all the previous time-steps. This results into the problem of vanishing or exploding gradients [14, 32]. A common way to mitigate the problem of vanishing or exploding gradients is to employ gated recurrent mechanisms such as the GRU (see (4)) [15].

[1] The functional dependence of variables and differentiable functions is used for convenience in the notation.

3. STATE SPACE INFORMED LOSS FUNCTIONS

In addition to the implicit teacher-forcing in the formulation of STNs, the work in [19] highlights the potential benefit of using state information as an additional objective to optimize neural networks. We refer to this objective as *state matching* and propose to use it for optimizing the parameters of a GRU. To that aim, let us denote by $\mathbf{s}_t \in \mathbb{R}^{N_s}$ the target vector of state variables at time step t where N_s is the number of measured states from the device under test and by $\mathbf{x}_t \in \mathbb{R}^N$ the output of the GRU using (4). The state matching loss E_s for a set of time steps $t \in [0, T-1]$, is given by

$$\text{E}_s = \sum_{t=0}^{T-1} \mathcal{L}_{\text{ESR},t}(\mathbf{x}_t, \mathbf{s}_t), \tag{10}$$

where $\mathcal{L}_{\text{ESR},t}$ is the error-to-signal ratio (ESR) loss function and it follows that $N = N_s$. The $\mathcal{L}_{\text{ESR},t}$ between a target $\mathbf{m}_t \in \mathbb{R}^N$ and an approximation of $\mathbf{m}_t$ at time-step t, denoted as $\hat{\mathbf{m}}_t \in \mathbb{R}^N$, is computed as

$$\mathcal{L}_{\text{ESR},t}\left(\mathbf{m}_t, \hat{\mathbf{m}}_t\right) = \frac{\|\mathcal{F}(\mathbf{m}_t) - \mathcal{F}(\hat{\mathbf{m}}_t)\|_2^2}{\|\mathcal{F}(\hat{\mathbf{m}}_t)\|_2^2 + \epsilon}, \tag{11}$$

where $\epsilon = 10^{-9}$ is an additive constant ensuring numerical stability and $\mathcal{F}$ is a memory-based pre-emphasis filter computed as

$$\mathcal{F}(\mathbf{m}_t) = \mathbf{m}_t - 0.95\mathbf{m}_{t-1}. \tag{12}$$

In (12) the memory refers to the storage of the information contained in $\mathbf{m}_{t-1}$ and for $t = 0$, $\mathbf{m}_{t-1}$ is a vector of zeros. The motivation for applying the high emphasis filter follows the experimental findings presented in [6, 17, 33].

Assuming that the outputs of the device under test are a subset of the measured states, (10) also contains information of the target output signal $\mathbf{y}$. In contrast, black-box training only aims to match outputs which is equivalent to substituting $\mathbf{x}_t$ and $\mathbf{s}_t$ in (10) by

$$\tilde{\mathbf{x}}_t = \left[\mathbf{0}_{Q\times(N-Q)}\ \mathbf{I}_{Q\times Q}\right]\mathbf{x}_t,\ \tilde{\mathbf{s}}_t = \left[\mathbf{0}_{Q\times(N-Q)}\ \mathbf{I}_{Q\times Q}\right]\mathbf{s}_t.$$

With the above substitution, we can interpret state matching as a form of loss *regularization*. If we were to naively use (10) to regularize GRU training however, this would impose that $N = N_s$ which could limit network capacity due to the restrictive form of f_g. Motivated by the different $\mathbf{A}_E$ matrices used in (3) and (7), we therefore propose a modified state matching loss by substituting $\mathbf{x}_t$ and $\mathbf{s}_t$ in (10) by

$$\tilde{\mathbf{x}}_t = \mathbf{A}_s\mathbf{x}_t + \mathbf{b}_s,\ \tilde{\mathbf{s}}_t = \mathbf{s}_t, \tag{13}$$

where $\mathbf{A}_s, \mathbf{b}_s$ together form a learnable affine mapping between the network states $\mathbf{x}_t$ and circuit states $\mathbf{s}_t$. This allows us to maintain the decoupling of N and N_s and optimize with respect to a single objective, i.e., Eq. (10) with the modifications imposed by Eq. (13). Furthermore, assuming that the output states form a subset of $\mathbf{s}$, the last Q rows of $\mathbf{A}_s$ and $\mathbf{b}_s$ are equivalent to the output affine mapping of the GRU, $\mathbf{W}_y$ and $\mathbf{b}_y$ respectively. We refer to the usage of (13) as *soft state matching* and the usage of (10) as *hard state matching*.

Expanding on (13), we can also use state information to warm-start the GRU during training. Commonly, for the first time-step in a batch, the GRU hidden states are initialized with zeros and for subsequent frames the hidden states contain the previously processed time-step state. In contrast, teacher forcing uses ground truth state information for every time-step. We therefore propose to use an additional learnable mapping between the state vector $\mathbf{s}_t$ and the GRU hidden states $\mathbf{x}_t$ during training, to complement the use of soft state matching. Specifically, at the start of each TBPTT frame in a batch we propose to set

$$\mathbf{x}_t = \mathbf{C}_s \mathbf{s}_t + \mathbf{d}_s, \tag{14}$$

where $\mathbf{C}_s$ and $\mathbf{d}_s$ constitute a learnable affine layer. Echoing our state matching naming convention, we will refer to this as *soft teacher forcing*. We postulate that when combined with soft state matching, this will allow GRU based networks to exploit circuit state information without restricting their state size.

4. EXPERIMENTAL PROCEDURE

To evaluate the impacts of teacher forced training and state-matching on NN VA modeling, we conducted a number of experiments. In this section we highlight these efforts by firstly outlining our experimental methodology including our data generation approach and performance metrics. For the teacher forcing experiments we explore the presence of exposure bias in trained models in the context of a second order diode clipper while in the state-matching case we consider two more complicated devices, a Boss SD-1 and MXR Phase 90. Lastly, listening tests results for each experiment are included to complement the objective performance metrics.

4.1. Data Generation

To conveniently obtain the state information of increasingly complex circuits, we opted to use synthetic data generated via SPICE simulations. A corpus of musical instrument recordings ($\sim$ 7 min) were passed through these models to form the dataset for each, with peak values of $\pm$1 V. The data was split into training ($\sim$ 5 min 30 s) and validation ($\sim$ 30 s) subsets. Furthermore, a test subset ($\sim$ 1 min) was formed to evaluate the trained model performance for each experiment. The number of states N_s was determined for each circuit based on their respective number of reactive elements (i.e., capacitors or inductors). As these states could be chosen somewhat arbitrarily without degrading performance, [19], for each reactive component, an associated nodal voltage was measured with the additional restriction that one such state would be the output signal itself (for all considered circuits $Q = 1$).

SPICE transient analysis uses a variable time step size to accurately resolve the circuit states. To convert into a constant step size, we therefore used cubic spline interpolation to obtain a 32 times oversampled sequence, which was then bandlimited and resampled to a 48 kHz rate[2]. Finally, the resulting recordings were scaled globally (per device) to a range of $\pm$1.

4.2. Objective Evaluation

While the models were trained with pre-emphasized ESR loss (11), each model was also evaluated on unseen test data using the ESR *without* pre-emphasis. The ESR metric without the pre-emphasis is denoted as $\mathrm{E}_{\mathrm{ESR}}$. To assess the model quality, we also employed a metric computed using the short-time Fourier transform (STFT) denoted by $\mathrm{E}_{\mathrm{STFT}}$. Similarly to (11), $\mathrm{E}_{\mathrm{STFT}}$ is computed as

$$\mathrm{E}_{\mathrm{STFT}} = \frac{1}{JK} \sum_{j=1}^{J} \sum_{k=1}^{K} 20 \log_{10} \left(\frac{|\mathrm{STFT}(\mathbf{m})|}{|\mathrm{STFT}(\hat{\mathbf{m}})|} \right), \tag{15}$$

where J and K denote the number of frames and frequency bins, respectively, $\log_{10}$ is an element-wise logarithmic function, and STFT denotes the STFT operation using a 8192 sample long Hanning windowing function and 75% overlap.

In contrast to the ESR, which compares values point-wise in the time domain, the STFT and magnitude based metric first averages short-time spectral information for the target and model output, and only afterwards computes a difference. The resulting metric is agnostic to discrepancies in phase. We expect a potential drift in model output to occur, but it remains imperceptible as long as the short-time spectral magnitude remains unaffected. Note that both the validation and test metrics were only considered on the final output of the device under study. This is due to the output accuracy being the most crucial element of the model performance, while allowing for the direct comparison of models irrespective of their state-space representation.

4.3. Impact of Exposure Bias on Inference Performance

To quantify exposure bias, the generalization gap between the free-running prediction performance, during inference, and the metrics measured during TBPTT based training was explored. Specifically, three model configurations were considered: the STN [19], a standard black-box GRU [17], and finally a state-space regularized GRU (SS-GRU) which utilizes the soft state-matching (13) and soft teacher forcing (14) proposed in Sec. 3. Furthermore, each model was trained using various TBPTT lengths to quantify the effect of sequential training. To keep the experiment simple, we focused on modeling a second order diode clipper (following [19]). To ensure non-linear circuit behavior, the input signals for each dataset were increased in order to give peak values of $\pm$2 V, which is comparable to the signals used in [19].

4.3.1. Experimental Configuration

The STN and SS-GRU were able to be trained in a fully teacher-forced manner, because the real circuit states were known at each time step and could be mapped to the model state. However, such ground-truth information was not available for the hidden states of the GRU. Therefore, when training the standard GRU with a TBPTT of 1 the model always starts from a zero state. Each model was trained with a constant number of hidden units, in this instance 8, which was chosen due to the simplicity of the given circuit. Each GRU model contained a single layer, while the STN had 3 hidden layers due to a fully connected MLP layer having approximately one third the computational complexity of a (three-gated) GRU layer of the same nominal size. Both the STN and SS-GRU map between the hidden and observed states using affine layers.

For each training iteration, a one-second audio segment was used which was framed and reshaped to match each respective TBPTT segment length. As a result, the effective minibatch size was decreased as TBPTT length increased, retaining a constant

[2]Note that the STN was originally trained at a 192 kHz rate [19].

Table 1: *Test set E_{ESR} and E_{STFT} metrics for the second order diode clipper with increasing TBPTT length (the lower the better), for both free-running and teacher-forced inference. The lowest obtained values in each column are highlighted in bold.*

Model	TBPTT	Free-running		Teacher-forced	
		E_{ESR} (%)	E_{STFT} (dB)	E_{ESR} (%)	E_{STFT} (dB)
STN	1	5.40	3.92	**0.18**	1.43
	4	6.77	2.44	6.83	2.46
	16	5.31	2.22	4.48	2.16
	64	5.14	2.09	4.55	1.92
	256	5.42	2.09	4.80	2.02
SS-GRU	1	8.85	3.59	0.45	0.93
	4	7.24	2.48	5.47	2.17
	16	1.49	1.57	0.32	0.89
	64	1.14	1.53	0.32	0.95
	256	**0.57**	**1.23**	**0.18**	**0.75**
GRU	1	19.24	4.21		
	4	5.11	2.14		
	16	4.31	1.92	N/A	N/A
	64	3.89	1.89		
	256	3.90	2.09		

data volume at each iteration. To evaluate the generalization performance during training, validation metrics were computed by performing free-running predictions over the full length sequences in the validation set every 10k iterations. The training utilized an early stopping criterion with a patience of 10 validation intervals, after which the model with the lowest validation loss was stored for further evaluation. We used the Adam optimizer [34] with a learning rate of 5×10^{-4} and beta values (0.9, 0.999) for all experiments in this paper. Likewise, the MLP layers were initialized using the Xavier uniform initialization [35], while the GRU used the PyTorch default uniform initialization.

4.3.2. *Exposure Bias Results*

Table 1 lists objective metrics for the various model architectures and TBPTT lengths used during training. TBPTT length 1 corresponds to the fully teacher-forced training for the state-space models. Overall, the SS-GRU tends to outperform the GRU model in matching size and TBPTT configuration. To quantify exposure bias, the state-space model scores were computed for the test set data first for free-running predictions on long sequences, and second for one-step predictions given ground truth state information. This was done in an attempt to separate the "ordinary generalization error" present in a teacher-forcing condition that matches the training setup and realistic generalization error present in the free-running mode. We argue that the difference between these two kinds of metric gives a measure of exposure bias in VA modeling.

Fig. 2 displays training and validation metrics for different models and various TBPTT lengths. It can be seen that fully teacher forced training, results in poor generalization when the models are validated in free-running mode. This is the fundamental outcome when a model is exposure biased, since free-running validation is the first instance wherein the model has to recursively utilize its own predictions. In contrast, as recursive conditioning is introduced to the training, the corresponding validation accuracy improves dramatically. That is because the models are optimized to operate in such conditions.

Furthermore, it is important to note the instability and poor generalization present when fully teacher forcing during training. However, this instability dissipates as recurrent predictions are introduced in training, highlighting the benefits of reducing exposure bias by having models train in circumstances that represent their inference conditions closest. Moreover, the generalization gap from teacher forced training to free-running validation is not as significant with the SS-GRU when comparing to the STN.

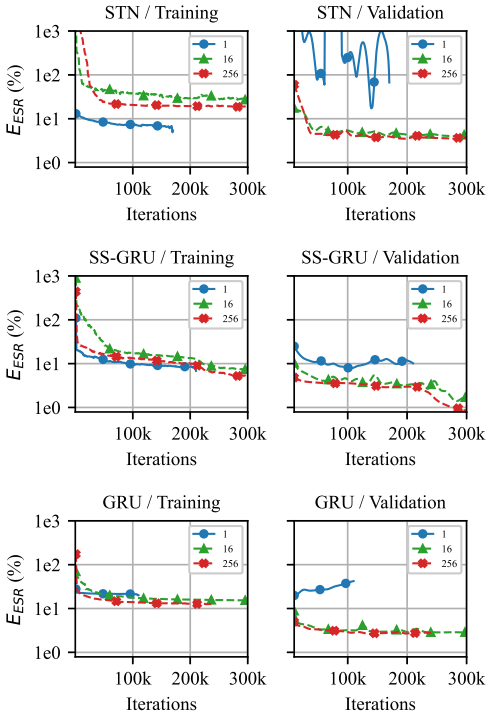

Figure 2: *E_{ESR} metric during training and free-running validation for each model type and configuration. Legend indicates the truncation order in TBPTT during training. ESR metric values are averaged for all model outputs during training (i.e., all states for state-space models, only output for standard GRU). During validation, the ESR metric is computed only on the final circuit output.*

From Fig. 2, one can observe a discrepancy between validation and training results for the majority of configurations. While for the state-aware models this comes from the fact that the validation metric only considers the output signal, the presence in the GRU results indicates potential differences in the datasets. In particular, the validation set is smaller and hence less diverse than that used during training. Lower validation scores may therefore indicate accurate modeling for only a subset of typical instrument playing.

4.4. State Matching with Neural Networks

To explore the effect of state-matching as a mechanism with which to train neural networks, we experimented with two devices of varying behavior, the Boss SD-1 and MXR Phase 90. For each device we tested both the state-aware architectures discussed within this work, the STN and SS-GRU, as well as the non-state matched GRU. For these experiments we adopted the training convention of previous works [17], whereby each iteration consisted of processing TBPTT length segments with a fixed batch size. For both devices the combination of a batch size of 4 with TBPTT length 256 was used, while the hidden unit size was increased.

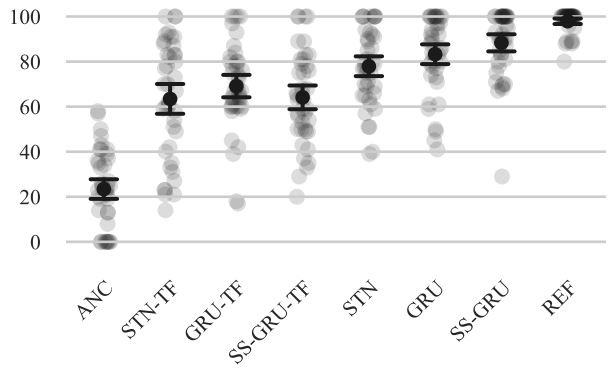

Figure 3: *Listening tests results for the diode clipper. For each model, the configurations with lowest free-running ESR (E_{ESR}) in Table 1 and TBPTT length 1 (denoted "TF") were considered.*

Table 2: *Test set* E_{ESR} *and* E_{STFT} *metrics for Boss SD-1. Lowest obtained values are highlighted in bold.*

Model	Hidden units	E_{ESR} (%)	E_{STFT} (dB)
STN	8	146.39	8.21
	16	47.03	13.25
	32	10.62	4.60
	64	48.01	12.48
SS-GRU	8	5.17	2.98
	16	1.97	2.14
	32	1.25	1.87
	64	0.85	1.52
GRU	8	7.66	3.19
	16	1.00	1.77
	32	0.79	1.46
	64	**0.53**	**1.25**

4.4.1. Boss SD-1

The Boss SD-1 is a common overdrive pedal in the world of guitar effects. Unlike the second order diode clipper, this circuit is much more complex and contains a number of tunable user controls. For the sake of brevity, these were set to a constant value but it should be noted that these could also have been passed in as additional inputs to the neural network to learn their behavior [17].

The results for the Boss SD-1 are shown in Table 2, where the increase in circuit complexity shows higher variance between models and configurations than before. For all configurations, the STN results in a higher loss value compared to an equally sized GRU, which highlights the challenges in training such a network architecture with TBPTT. Both GRUs demonstrate improved performance with increasing hidden size, however, when comparing the smallest model sizes, one can note that the SS-GRU outperforms its GRU counterpart. This could indicate that access to circuit state-space during training allows for more efficient performance with smaller hidden sizes. The SS-GRU can also leverage the warm-starting of the circuit state to accumulate less error within a batch. The state-matching also implicitly demonstrates longer-term dependencies that would otherwise fall outside of the TBPTT window. However, the larger hidden sizes allow the standard GRU to perform best, which could demonstrate a loss of model capacity of the SS-GRU as it has to replicate each system state of the device.

Table 3: *Test set* E_{ESR} *and* E_{STFT} *metrics for the MXR Phase 90. Lowest obtained values are highlighted in bold.*

Model	Hidden units	E_{ESR} (%)	E_{STFT} (dB)
STN	8	57.09	8.30
	16	58.02	8.31
	32	56.36	8.08
	64	56.33	7.98
SS-GRU	8	8.33	2.78
	16	3.96	2.35
	32	**1.04**	**1.45**
	64	1.27	**1.45**
GRU	8	29.47	4.03
	16	6.27	2.26
	32	3.89	1.74
	64	1.15	1.48

4.4.2. MXR Phase 90

The final circuit considered is that of the MXR Phase 90 which is a basic four pole phaser pedal. Unlike the other models considered so far, the input vector $\mathbf{u}_n$ in this case is given by the concatenation of the input voltage V_{in} and low frequency oscillator (LFO) voltage V_{LFO} such that $\mathbf{u}_n = \left[V_{in,n},\ V_{LFO,n}\right]^T$. Notably, V_{LFO} is a modulation signal used to vary the center frequency of the phasers allpass stages across time. Since the device was simulated within SPICE, the LFO section of the physical device was not replicated but rather a triangular waveform was used to modulate the allpass filter stages within realistic bounded values. The frequency of oscillation was set to different values for each dataset, to effectively evaluate the generalization capabilities of the models. Since this allowed the ground truth time-aligned LFO signal to be known directly, no LFO extrapolation was required as shown in [36].

The objective metric scores for the Phase 90 experiments can be seen in Table 3. In contrast to the previous devices, the SS-GRU now provided the most accurate emulation. Moreover, similar to the SD-1 (see Table 2), the SS-GRU performed better with smaller hidden sizes. The differences between the GRU models is also shown in Fig. 4, in which spectrograms of the smallest and best performing models are included. Again, the STN exhibits higher E_{ESR} values when compared to equally sized GRUs, reinforcing the training challenges noted in Sec. 4.4.1.

4.5. Listening Tests

To grade the subjective performance of the studied models, we conducted a set of MUSHRA listening tests[3] [37]. Fig. 3 shows the results for the diode clipper experiments, while Fig. 5 displays the results for the SD-1 and Phase 90. Rating averages are shown with t-statistic based 95% confidence intervals, along with scatter plots of individual ratings. Test responses where the reference was rated below 80 or the anchor was rated above 80 were excluded from the analysis.

The Wilcoxon signed-rank test was used to evaluate the statistical significance of differences between systems. For the significance test, all pairings (excluding the anchor) were formed within a MUSHRA page and the resulting p-values were adjusted for multiple comparisons using the Bonferroni correction. For the diode clipper, there were statistically significant differences between all conditions, except for the pairings STN-TF – GRU-TF, STN-TF –

[3] Audio examples available at https://neural-dsp-publications.github.io/DAFx2021/

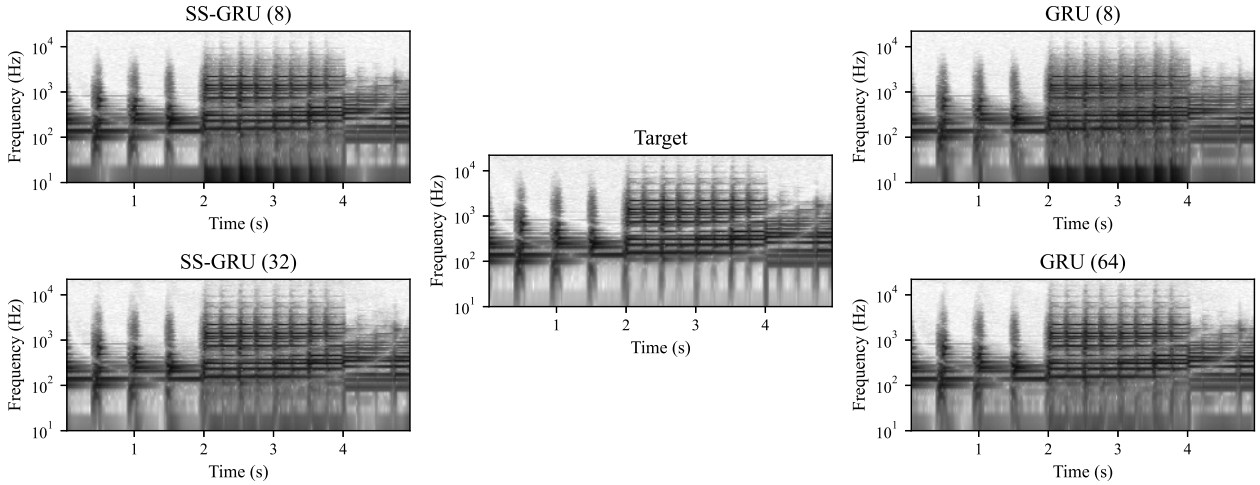

Figure 4: *Phase 90 test set spectrogram for GRU models of increasing size. Model's hidden size is indicated in parenthesis.*

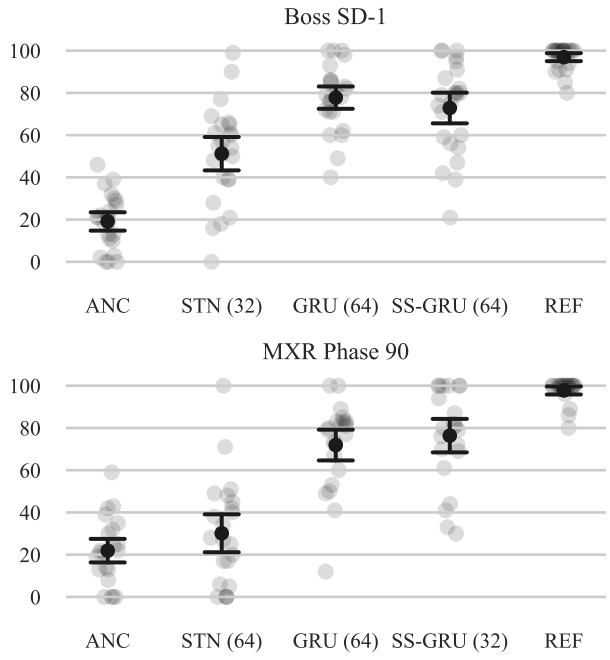

Figure 5: *Listening tests results for the (top) SD-1 and (bottom) Phase 90 models. Model's hidden size is indicated parenthesis.*

SS-GRU-TF, STN-TF – STN, GRU-TF – SS-GRU-TF, GRU-TF – STN, STN – GRU, GRU – SS-GRU. For the SD-1 and Phase 90, there were statistically significant differences between all conditions, except for the pairing GRU – SS-GRU.

5. CONCLUSIONS

This paper presented a framework for interpreting STNs and GRUs as special cases of a learnable DK-method. The framework enabled us to investigate two questions: first, do such models suffer from exposure bias in a measurable way, and second, does matching of circuit states to model states assist in NN training. These questions were motivated by previous knowledge in RNN research (especially language modeling) and VA modeling, respectively. Both the objective metrics and listening tests show that all the studied model configurations exhibit a test-set performance gap between free-running and teacher-forced inference, which we interpret as exposure bias. The effect is somewhat mitigated by TBPTT during training, but full understanding of the issue remains a future research question. State matching for the GRU was found helpful in the case of a phaser circuit (Phase 90) and diode clipper, but not with a more complex distortion circuit (SD-1).

While the STN performed reasonably well on the diode clipper and benefited from the TBPTT training, it struggled to model the more complicated SD-1 and Phase 90 circuits. Although there is no available experimental reference for these circuits, we acknowledge that implementation differences to the original paper [19] may have contributed to this performance. Notably, all the experiments in this paper used a 48 kHz sample rate, while the original used 192 kHz. Furthermore, we did not perform an extensive hyperparameter search for the model and optimizer configurations, nor experimented using different parameter initialization strategies. The settings used were rather conservative but remained consistent across the different models considered. While the models used herein were fairly small, some sensitivity can be expected. However, investigating these differences is left as future work.

6. REFERENCES

[1] V. Välimäki, S. Bilbao, J. O. Smith, *et al.*, "Virtual analog effects," in *DAFX: Digital Audio Effects*, U. Zölzer, Ed., Second, Chichester, UK: Wiley, 2011, pp. 473–522.

[2] J. Pakarinen and D. T. Yeh, "A review of digital techniques for modeling vacuum-tube guitar amplifiers," *Computer Music J.*, vol. 33, no. 2, pp. 85–100, 2009.

[3] S. D'Angelo, "Lightweight virtual analog modeling," in *Proc. 22nd Colloquium on Music Informatics*, 2018, pp. 59–63.

[4] F. Eichas and U. Zölzer, "Black-box modeling of distortion circuits with block-oriented models," in *Proc. DAFx*, Brno, Czech Republic, Sep. 2016, pp. 39–45.

[5] ——, "Gray-box modeling of guitar amplifiers," *J. Audio Eng. Soc.*, vol. 66, no. 12, pp. 1006–1015, Dec. 2018.

[6] E.-P. Damskägg, L. Juvela, E. Thuillier, and V. Välimäki, "Deep learning for tube amplifier emulation," in *Proc. ICASSP*, Brighton, UK, May 2019, pp. 471–475.

[7] E.-P. Damskägg, L. Juvela, and V. Välimäki, "Real-time modeling of audio distortion circuits with deep learning," in *Proc. SMC*, May 2019, pp. 332–339.

[8] M. Martìnez Ramìrez and J. Reiss, "Modeling nonlinear audio effects with end-to-end deep neural networks," in *Proc. ICASSP*, Brighton, UK, May 2019, pp. 171–175.

[9] J. Engel, L. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable digital signal processing," in *Proc. ICLR*, 2020.

[10] B. Kuznetsov, J. D. Parker, and F. Esqueda, "Differentiable IIR filters for machine learning applications," in *Proc. DAFx*, Vienna, Austria, Sep. 2021, pp. 297–303.

[11] S. Nercessian, A. Sarroff, and K. J. Werner, "Lightweight and interpretable neural modeling of an audio distortion effect using hyperconditioned differentiable biquads," in *Proc. ICASSP*, Toronto, Canada, Jun. 2021, pp. 890–894.

[12] D. T. Yeh, J. S. Abel, and J. O. Smith, "Automated physical modeling of nonlinear audio circuits for real-time audio effects—Part I: Theoretical development," *IEEE Trans. Audio Speech Lang. Processing*, vol. 18, no. 4, pp. 728–737, 2010.

[13] F. Fontana and F. Avanzini, "Computation of delay-free nonlinear digital filter networks: Application to chaotic circuits and intracellular signal transduction," *IEEE Trans. on Signal Processing*, vol. 56, no. 10, pp. 4703–4715, 2008.

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

[15] K. Cho, B. Van Merriënboer, C. Gulcehre, *et al.*, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proc. Empirical Methods in Natural Language Processing*, 2014.

[16] T. Schmitz and J.-J. Embrechts, "Nonlinear real-time emulation of a tube amplifier with a long short time memory neural-network," in *Proc. AES 144th Convention*, 2018.

[17] A. Wright, E.-P. Damskägg, and V. Välimäki, "Real-time black-box modelling with recurrent neural networks," in *Proc. DAFx*, Birmingham, UK, Sep. 2019, pp. 173–180.

[18] A. Wright, E.-P. Damskägg, L. Juvela, and V. Välimäki, "Real-time guitar amplifier emulation with deep learning," *Applied Sciences*, vol. 10, no. 3, Feb. 2020.

[19] J. D. Parker, F. Esqueda, and A. Bergner, "Modelling of nonlinear state-space systems using a deep neural network," in *Proc. DAFx*, Birmingham, UK, Sep. 2019, pp. 165–172.

[20] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, *et al.*, Eds., vol. 28, Curran Associates, Inc., 2015.

[21] S. Wiseman and A. M. Rush, "Sequence-to-sequence learning as beam-search optimization," in *Proc. 2016 Conference on Empirical Methods in Natural Language Processing*, Nov. 2016, pp. 1296–1306.

[22] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba, "Sequence level training with recurrent neural networks," in *Proc. ICLR*, 2016.

[23] A. Goyal, A. Lamb, Y. Zhang, *et al.*, "Professor forcing: A new algorithm for training recurrent networks," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, *et al.*, Eds., vol. 29, 2016.

[24] L. Yu, W. Zhang, J. Wang, and Y. Yu, "SeqGAN: Sequence generative adversarial nets with policy gradient," in *Proc. 31st AAAI Conference on Artificial Intelligence*, AAAI Press, 2017, pp. 2852–2858.

[25] F. Schmidt, "Generalization in generation: A closer look at exposure bias," in *Proc. 3rd Workshop on Neural Generation and Translation*, Association for Computational Linguistics, Nov. 2019, pp. 157–167.

[26] P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[27] A. Peussa, "State-space virtual analogue modelling of audio circuits," M.S. thesis, Aalto University, 2020.

[28] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991.

[29] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Computation*, vol. 1, no. 2, pp. 270–280, 1989.

[30] M. I. Jordan, "Attractor dynamics and parallelism in a connectionist sequential machine," in *Artificial Neural Networks: Concept Learning*. IEEE Press, 1990, pp. 112–127.

[31] A. Graves, "Supervised Sequence Labelling with Recurrent Neural Networks," Ph.D. dissertation, Technical University Munich, 2008.

[32] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proc. ICML*, 2013, pp. 1310–1318.

[33] A. Wright and V. Välimäki, "Perceptual loss function for neural modeling of audio systems," in *Proc. ICASSP*, Barcelona, Spain, May 2020, pp. 251–255.

[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015.

[35] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. 13th international conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[36] A. Wright and V. Välimäki, "Neural modelling of LFO modulated time varying effects," in *Proc. DAFx*, Vienna, Austria, Sep. 2020, pp. 281–288.

[37] M. Schoeffler, S. Bartoschek, F.-R. Stöter, *et al.*, "WebMUSHRA – A comprehensive framework for web-based listening tests," *J. Open Research Software*, vol. 6, no. 1, Feb. 2018.