Pruning Deep Neural Network Models of Guitar Distortion Effects

Südholt, David ; Wright, A.; Erkut, Cumhur; Välimäki, Vesa

Published in:
IEEE/ACM Transactions on Audio, Speech, and Language Processing

DOI (link to publication from Publisher):
10.1109/taslp.2022.3223257

Publication date:
2023

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

Pruning Deep Neural Network Models of Guitar Distortion Effects

David Südholt, Alec Wright, Cumhur Erkut, and Vesa Välimäki, *Fellow, IEEE*

***Abstract*—Deep neural networks have been successfully used in the task of black-box modeling of analog audio effects such as distortion. Improving the processing speed and memory requirements of the inference step is desirable to allow such models to be used on a wide range of hardware and concurrently with other software. In this paper, we propose a new application of recent advancements in neural network pruning methods to recurrent black-box models of distortion effects using a Long Short-Term Memory architecture. We compare the efficacy of the method on four different datasets; one distortion pedal and three vacuum tube amplifiers. Iterative magnitude pruning allows us to remove over 99% of parameters from some models without a loss of accuracy. We evaluate the real-time performance of the pruned models and find that a 3x-4x speedup can be achieved, compared to an unpruned baseline. We show that training a larger model and then pruning it outperforms an unpruned model of equivalent hidden size. A listening test confirms that pruning does not degrade the perceived sound quality, but may even slightly improve it. The proposed techniques can be used to design computationally efficient deep neural networks for processing the sound of the electric guitar in real time.**

***Index Terms*—Audio systems, machine learning, music, supervised learning, recurrent neural networks.**

I. Introduction

DISTORTION is one of the most common effects applied to the signal of an electric guitar. Many popular devices that achieve this effect, such as amplifiers and effects pedals, are based on analog circuitry. These circuits are characterized by their use of nonlinear components such as transistors, diodes, and vacuum tubes.

In virtual analog (VA) modeling, the aim is to reproduce the behavior of such devices in software as accurately as possible, accounting for the real-time constraints [1], [2], [3]. This allows the effects to be used in a digital audio workstation (DAW) as part of a fully digital music production workflow, reducing the need to purchase, store and transport analog equipment.

Approaches to VA modeling of distortion circuits can broadly be classified as *white-box*, *black-box*, or *gray-box* [2]. In white-box modeling [4], [5], [6], [7], the circuitry is analyzed and simulated using time-stepping methods. Additionally, these models can then be further optimized using data recorded from the target device [8], [9]. This can achieve very accurate results, but often involves a labor-intensive design process and can produce computationally demanding models.

In black-box modeling, the circuit's response to some input signals is measured, and the model aims to recreate the observed input-output mapping through various means. Those include dynamical convolution [10], the Volterra series [11], [12], or Wiener models, combining a linear filter with a static nonlinearity [13], [14]. The strength of black-box models lies in their generality, and the same model design can be used to emulate a range of different systems. However, black-box models typically fail to reach the same accuracy as white-box models, and either do not model user controls at all or require the input-output measurements to be performed at multiple different configurations.

Gray-box modeling [15], [16], [17], [18] falls somewhere between white- and black-box approaches. The model is designed using knowledge of the circuit, but still requires input-output measurements of the emulated device. Modeling the input-output mappings with machine learning techniques has become popular in recent years. Gray-box approaches that model the circuit's states and output using kernel regression [15] or a deep neural network [17] have been proposed.

Various neural network models have also seen increasing use in virtual analog modeling. Zhang et al. introduced a many-layered long short-term memory (LSTM) network architecture in [19], and a single-layered LSTM along with a real-time C++ implementation was presented in [20]. Various recurrent, dense and convolutional models are analyzed in [21]. Damskägg et al. [22] proposed a convolutional model based on the WaveNet [23] architecture and presented a real-time implementation [24]. While they focused on the emulation of distortion pedals [24], Wright et al. compared performance of recurrent and convolutional models when applied to the emulation of vacuum tube amplifiers [25].

Real-time performance is an important aspect of VA modeling. Improving the processing speed and memory footprint of a model can allow it to run with lower latency, on a wider range of hardware, and concurrently with other, possibly

Manuscript received 15 December 2021; revised 27 October 2022; accepted 8 November 2022. Date of publication 18 November 2022; date of current version 2 December 2022. This work was supported by the Nordic Sound and Music Computing Network—NordicSMC," NordForsk under Grant 86892. This work was conducted during the research visit of David Südholt to the Aalto Acoustics Lab in September and December 2021. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Hema A Murthy. *(Corresponding author: David Südholt.)*

David Südholt was with the CREATE, Aalborg University Copenhagen, Copenhagen, Denmark. He is now with the Centre for Digital Music, Queen Mary University of London, London, U.K. (e-mail: david.suedholt@gmail.com).

Alec Wright and Vesa Välimäki are with the Acoustics Laboratory, Department of Signal Processing and Acoustics, Aalto University, 02150 Espoo, Finland (e-mail: alec.wright@aalto.fi; vesa.valimaki@aalto.fi).

Cumhur Erkut is with the CREATE, Aalborg University Copenhagen, Copenhagen, Denmark (e-mail: cer@create.aau.dk).

Digital Object Identifier 10.1109/TASLP.2022.3223257

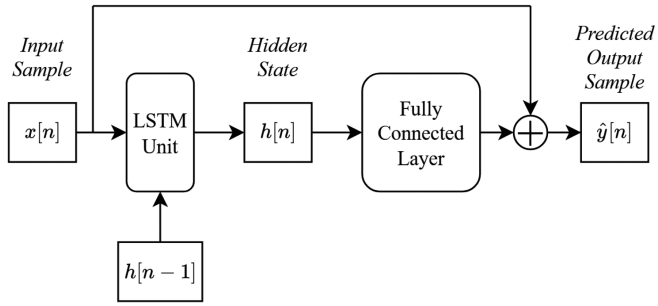

Fig. 1. Deep neural network architecture studied in this work [40].

resource-intensive plugins. Recent work has combined aspects of deep learning with traditional IIR filters to produce more efficient and interpretable models [26], [27], whilst Steinmetz and Reiss have explored efficient convolutional models for real-time VA modeling [28].

The inference process of many neural networks can be made more efficient through pruning away weights [29], [30]. Various pruning methods have been proposed that are specific to general-purpose convolutional [31], [32], [33] or recurrent [34], [35], [36] neural network architectures.

The Lottery Ticket Hypothesis (LTH) [37] conjectures that dense feed-forward networks contain subnetworks, referred to as winning tickets, that perform with comparable accuracy to the original network when trained in isolation. These subnetworks can be uncovered by iteratively pruning and retraining the network. Esling et al. [38] confirmed the existence of winning tickets for popular generative audio network architectures, such as the convolutional WaveNet [23] and the recurrent DDSP autoencoder [39].

This paper investigates the real-time performance improvements that can be gained from applying LTH-based pruning techniques to the LSTM distortion model proposed in [20]. We apply the models to the emulation of the Electro-Harmonix Big Muff Pi distortion pedal, the Blackstar HT-1 and HT-5 Metal amplifiers, and the Mesa Boogie 5:50 Express Plus amplifier.

The rest of the paper is organized as follows. Section II describes how LSTM models can be used for black-box VA modeling. In Section III, we present the approach to pruning the model. Section IV details the setup of our experiments to determine the effect of the pruning on accuracy and real-time performance, and Section V presents their results. Section VI concludes the paper.

II. Recurrent Neural Networks for Black-Box Modelling

The recurrent model discussed in [25] produces a single output sample based on a single input sample at each time step. It consists of a single LSTM unit, the output of which is fed into a fully connected (FC) layer, as shown in Fig. 1. A residual connection adds the input sample value to the output of the FC layer at each time step. This means that the desired output of the FC layer is the difference between the input sample and the corresponding sample that the emulated device would output at this time step. The rest of this section will explore the model in detail.

A. Long Short-Term Memory Layer

The key characteristic of RNNs is their statefulness. At each time step, the RNN output depends on its state at the previous time step, allowing it to model time series data while processing only one sample at a time.

A particularly popular method of keeping an internal state is the LSTM unit, first proposed in [41]. The LSTM state is composed of the *hidden state*, h, and the *cell state*, c. At time step n, the LSTM uses its previous states $h[n-1]$ and $c[n-1]$, along with the current input sample $x[n]$, to produce the outputs $h[n]$ and $c[n]$ according to the following set of functions:

$$i[n] = \sigma(W_{ii}x[n] + b_{ii} + W_{hi}h[n-1] + b_{hi}), \tag{1a}$$

$$f[n] = \sigma(W_{if}x[n] + b_{if} + W_{hf}h[n-1] + b_{hf}), \tag{1b}$$

$$\tilde{c}[n] = \tanh(W_{ic}x[n] + b_{ic} + W_{hc}h[n-1] + b_{hc}), \tag{1c}$$

$$o[n] = \sigma(W_{io}x[n] + b_{io} + W_{ho}h[n-1] + b_{ho}), \tag{1d}$$

$$c[n] = f[n]c[n-1] + i[n]\tilde{c}[n], \tag{1e}$$

$$h[n] = o[n]\tanh(c[n]), \tag{1f}$$

where $i[n]$ is the *input gate*, $f[n]$ the *forget gate*, $\tilde{c}[n]$ the *candidate cell state*, $o[n]$ the *output gate*, $\tanh(\cdot)$ the hyperbolic tangent function and $\sigma(\cdot)$ the logistic sigmoid function. The weight matrices W and biases b are learnable parameters. In this model, the vectors h and c are both of the same size S_{hid}, referred to as the *hidden size* of the LSTM. The total number of learnable parameters N_{param} of the LSTM can be expressed as

$$N_{\text{param}} = (4S_{\text{in}} + 8)S_{\text{hid}} + 4S_{\text{hid}}^2, \tag{2}$$

where S_{in} denotes the input size. In the simple case where the input consists of a single sample of mono-channel audio, $S_{\text{in}} = 1$.

B. Fully Connected Layer

The input of the FC layer is the hidden state $h[n]$ of the LSTM. Since this model does not use a nonlinear activation function in the FC layer, its output can be simply viewed as an affine transformation of the LSTM hidden state. Including the residual connection, the predicted output $\hat{y}[n]$ of the model is given by:

$$\hat{y}[n] = W_{fc}h[n] + b_{fc} + x[n]. \tag{3}$$

Since the output is again a single sample of mono-channel audio, the weight W_{fc} is simply a vector of size equal to the LSTM hidden size, and the bias b_{fc} is a single scalar, which can be seen as a learned dc offset.

III. Compression of Recurrent Neural Networks

In this section, we describe the method we employ to reduce the size of the model in order to speed up inference without sacrificing accuracy in the process.

A. Iterative Magnitude Pruning

Removing parts of a fully trained network leads to an initial loss in accuracy, which can be recovered by retraining the pruned model [30]. Repeating the pruning and retraining steps iteratively until a target sparsity is reached or until accuracy starts

to degrade generally results in higher accuracy than pruning in *one shot* [30], [37], [42].

When considering a neural network as a function $f(x;\theta)$ with input x and network parameters θ, pruning can be formalized as introducing a mask $m \in \{0,1\}^{|\theta|}$ [37]. Removing a weight is realized by setting its corresponding mask entry to zero, and the network function becomes $f(x;\theta \odot m)$, where $\odot$ denotes the element-wise product.

In *unstructured global pruning*, all of the parameters are pooled together, assigned a score, and the $p\%$ of the weights with the lowest score are removed for some pruning ratio p. In *magnitude pruning*, that score is simply equal to the weight's magnitude. Esling et al. [38] evaluated a number of other scoring methods for pruning generative audio models and found that magnitude pruning preserves the most accuracy in higher pruning contexts ($p > 90$).

Before the introduction of the LTH [37], the most common retraining technique was to fine-tune the model after pruning for a number of epochs at a fixed learning rate [30], [43]. With the proposal of the LTH came the introduction of *weight rewinding*, in which the remaining weights after pruning are reset to their values at some early epoch or initialization. Renda et al. [42] propose that *learning rate rewinding*, in which the weights are kept but the learning rate schedule is reset to its initialization, generally matches or outperforms weight rewinding, and suggest it as a network-agnostic default choice for pruning.

B. Early Stopping Based on Mask Distance

Reaching a desired level of sparsity by fully training a model and then applying iterative pruning, fully retraining the model at each iteration, incurs a large computational cost. Since it can be observed that important connectivity patterns emerge early in training and become relatively fixed in later stages [44], it has been proposed that so-called Early-Bird (EB) winning tickets can be identified at a very early training stage [45].

The valuable output of a given pruning iteration is not necessarily a fully trained, accurate model, but rather the pruning mask on which the next iteration of pruning is based. This allows us to stop the pruning iteration early when the mask is identified.

The decision to stop training is based on calculating a pruning mask at each epoch (without actually applying it) and monitoring their pair-wise Hamming distance. Once the pruning masks cease to change by more than a chosen threshold from epoch to epoch, the EB ticket is considered identified. The most recent pruning mask is applied and the next iteration is started. Training is stopped once the most recent k mask distances all fall below some threshold ϵ for some small constant k, to account for fluctuation in the early training stages.

Algorithm 1 describes the combination of unstructured global iterative magnitude pruning and the EB early stopping criterion that we use to obtain a pruned model. In a variant of this algorithm referred to as Fully Train First (FTF), the EB early stopping criterion is not applied in the first iteration of the algorithm, instead training the unpruned model for a fixed number of epochs.

Algorithm 1: The Mask Search Algorithm.

Initialize parameters θ, pruning mask m, learning rate scheduler l, pruning ratio p, number of pruning iterations N, mask distance threshold ϵ, and a FIFO queue Q with length k
for $i = 1$ to N **do**
  Set epoch counter $t = 0$
  while $\max(Q) < \epsilon$ **do**
    Train one epoch, updating l and θ where $m \neq 0$
    Calculate pruning mask m_t by pruning $p\%$ of currently active parameters θ where $m \neq 0$
    Calculate mask distance between m_t and m_{t-1} and add to Q
    $t = t + 1$
  end while
  Set $m = m_t$
  Set $\theta = \theta \odot m$
  Reset learning rate scheduler l
end for
return θ, m

C. Model Compression

Simply sparsifying the weight matrices through pruning does not result in notable performance improvements during inference. In [35], the authors find that for large LSTMs ($S_{\text{hid}} = 1500$, $S_{\text{inp}} = 1500$), the sparse matrix implementation of the Intel Math Kernel Library adds overhead and actually slows down inference at sparsities less than 80%. This matches our preliminary experiments with unstructured sparsity in our much smaller VA models ($S_{\text{hid}} \leq 96$, $S_{\text{inp}} = 1$), using the sparse matrix functionality of the Eigen library in our real-time implementation.

A more significant speedup can be achieved by exploiting the *intrinsic sparse structures* (ISS) that were shown to exist in LSTMs in [35]. Applying unstructured global magnitude pruning to the weight matrices of LSTMs gravitates towards masking entire related rows, so that some hidden states of the LSTM become useless. They can then be entirely removed from the model without affecting the computation graph at all, leading to an effective reduction in hidden size, which has a much more direct impact on inference performance. From here on, we will refer to the number of hidden nodes that are still active in a pruned model as the *effective hidden size*.

IV. Experimental Setup

We perform a number of experiments to investigate to what degree the LSTM black-box VA model can be compressed without sacrificing accuracy, and how this benefits real-time performance. Training deep neural networks depends on a large number of design decisions and hyperparameters, which this section describes in detail.

A. Training Data

1) Data Acquisition: We train the models on four sets of training data. Two of these datasets, the Big Muff pedal and

the HT-1 amplifier, are identical to those used in [20]. The 8 min and 10 s of input audio are sourced from the guitar and bass guitar datasets presented in [46], [47]. The dataset is split into a training set of 5 min and 42 s, a validation set of 1 min and 24 s and a test set of 1 min and 4 s. All audio is sampled at a standard rate of 44.1 kHz.

For the HT-1 amplifier and the Big Muff pedal, the output data is obtained by feeding the input data into the modeled devices using a MOTU UltraLite-mk3 USB audio interface. Both the output of the audio interface and the output of the modeled device are recorded to construct the input-output data used in training.

For the HT-5 and Mesa amplifiers, we use data from an existing dataset [48]. The training set consists of 2 min and 43 s of audio, which was previously shown to be sufficient to train a perceptually accurate model on these devices [25].

2) Modeled Devices: The Electro-Harmonix Big Muff guitar pedal (Big Muff) is a famous distortion effect and has been a popular subject of VA modeling research [13], [24], [49]. It achieves a heavily distorted sound with two cascaded diode clipping stages. The pedal offers a "sustain" and "volume" knob that control the pre- and post-gain, respectively, as well as a "tone" knob to control the shape of a combined lowpass and highpass filter after the clipping stages.

The Blackstar HT-1 (HT-1) is a 1 W vacuum tube amplifier with a high-gain and a low-gain channel. In addition to controls for pre- and post-gain, it offers a knob controlling the "Infinite Shape Feature" (ISF). According to the manufacturer, it allows for continuous shifting between two distinct tone settings, described as the "American" tone when turned fully counterclockwise and the "British" tone when turned fully clockwise.

The Blackstar HT-5 Metal (HT-5) offers similar controls to the HT-1 amplifier, but has a wattage of 5 W and distorts more heavily on both channels. Since the focus of this work is on modeling distortion, recordings from the high-gain channel are used for both Blackstar amplifiers.

The Mesa Boogie 5:50 Express Plus (Mesa) is also a 5W tube amplifier, offering the choice between channels labeled "Clean or Crunch" and "Blues or Burn". Recordings are obtained from the "Crunch" channel.

B. Model Training

During all training runs, the models are trained using the Adam optimizer [50] with an initial learning rate of 5×10^{-4}. Additionally, the training loss function is calculated on the validation set every second training epoch, and the learning rate is halved if the loss does not improve for five consecutive validations.

The models and experiments were implemented using the PyTorch [51] and PyTorch Lightning [52] libraries. The code and the raw and processed audio used for training is available online[1].

1) Loss Function: We train the model to minimize the loss function from [20], which is a combination of the error-to-signal ratio (ESR) loss used in [22] and [24], and an additional dc loss. Before computing the ESR loss, a first-order high-pass filter with the transfer function

$$H(z) = 1 - 0.85z^{-1} \tag{4}$$

is applied to the output. This pre-emphasis filter helps the network learn to model the high-frequency content [40]. Giving a segment of training signal of length N, the pre-emphasized ESR is given by:

$$\mathcal{E}_{\text{ESR}} = \frac{\sum_{n=0}^{N-1} |y_p[n] - \hat{y}_p[n]|^2}{\sum_{n=0}^{N-1} |y_p[n]|^2}, \tag{5}$$

where y_p is the pre-emphasized target signal, and $\hat{y}_p$ is the pre-emphasized neural network output.

The dc loss term is given by:

$$\mathcal{E}_{\text{dc}} = \frac{\left| \frac{1}{N} \sum_{n=0}^{N-1} (y[n] - \hat{y}[n]) \right|^2}{\frac{1}{N} \sum_{n=0}^{N-1} |y[n]|^2}. \tag{6}$$

The final loss function used in training is the weighted combination of the ESR and dc loss given by:

$$\mathcal{E} = 0.75\mathcal{E}_{\text{ESR}} + 0.25\mathcal{E}_{\text{dc}}. \tag{7}$$

2) Truncated Backpropagation Through Time: Updating the parameters of the LSTM after every time step comes at a prohibitive computational cost for audio sequences, since that would result in a number of parameter updates per second of training data equal to the sampling rate. *Backpropagation through time* (BPTT) [53] refers to a method of training RNNs in which an entire training sequence is processed before updating the network parameters. This is also not well-suited to audio data consisting of a large number of time steps, since backpropagating through long sequences is also computationally expensive and the infrequent parameter updates require the sequences to be processed many times, making training very slow.

Truncated BPTT [54] is a compromise between the two methods, in which parameter updates are carried out during the processing of a sequence. The recurrent unit state persists between updates. We train our models with truncated BPTT, carrying out parameter updates every 2048 samples, roughly corresponding to 50 ms of audio.

3) Batch Processing: The training dataset is split into segments of 22050 samples, corresponding to half a second of audio. During training, the segments are processed in mini-batches of 40 segments, the order of which is randomized at the beginning of each training epoch.

At the start of each mini-batch, the LSTM state is set to 0. The first 1000 samples of each segment are treated as "warmup" samples, which are processed without performing parameter updates, allowing the LSTM state to initialize. Afterwards, parameters are updated according to the previously described process of truncated BPTT, resulting in roughly 10 parameter updates per segment.

The validation and test datasets are neither segmented nor pre-emphasized and simply processed as-is.

4) Mask Search: We perform 15 iterations of the mask search described in algorithm 1, pruning 30% of the weights in each iteration, resulting in a maximal sparsity of 99.53%. The pruning

[1][Online]. Available: https://github.com/dsuedholt/PrunedGuitarVA

TABLE I
PERFORMANCE OF UNPRUNED LSTM MODELS AND AT SELECTED STAGES OF PRUNING. THE BEST RESULTS ARE HIGHLIGHTED

Device	Hidden Size		# of Parameters		Pruned	Test Loss		Real-Time Factor	
	Unpruned	Pruned	Unpruned	Pruned	Weights [%]	Unpruned	Pruned	Unpruned	Pruned
Big Muff	32	32	4480	753	88.2	7.3	7.0	0.091	0.089
	64	**20**	17152	390	**99.6**	5.5	5.4	0.184	**0.057**
	96	24	38016	**369**	99.5	4.4	**4.2**	0.223	0.060
HT-1	32	**32**	4480	**1270**	76.0	3.5	3.9	0.108	**0.100**
	64	60	17152	2438	88.2	1.7	1.7	0.206	0.173
	96	76	38016	3676	**91.8**	**1.0**	1.2	0.246	0.187
Mesa	32	26	4480	**452**	94.2	0.61	0.59	0.102	0.079
	64	29	17152	903	96.0	0.31	0.34	0.195	0.083
	96	**24**	38016	929	**98.0**	0.22	**0.21**	0.219	**0.072**
HT-5	32	**32**	4480	**1705**	65.7	6.0	5.8	0.111	**0.105**
	64	54	17152	4427	76.0	3.8	3.9	0.210	0.127
	96	57	38016	2603	**94.2**	**1.2**	1.4	0.249	0.165

iteration is stopped when the mask distance from training epoch to training epoch is less than $\epsilon = 0.1$ for $k = 5$ consecutive epochs.

C. Speed Measurements

The impact of model compression and sparsity on inference performance is measured using the real-time JUCE implementation presented in [20]. We process a second of audio three times, measuring the processing time in each run and averaging the results. The speed measurements are reported in the form of the *real-time factor* (RTF), which is the ratio between the required processing time and the duration of the processed audio. An RTF value smaller than 1.0 means that the model runs in real time.

V. RESULTS

In our experiments, we investigate the impact that the mask search algorithm has on the accuracy and real-time performance of LSTM models of varying hidden sizes. Table I shows an overview of the results on the different datasets obtained with the FTF variant of the mask search algorithm.

On the Mesa and Big Muff datasets, applying the mask search algorithm to larger model results in a pruned model that outperforms smaller base models both in accuracy and real-time performance. The pruning speeds up the largest model by a factor of 3.1 (Mesa) and 3.7 (Big Muff).

On these datasets, training a larger model of hidden size 96 and then pruning it yields models whose effective hidden size is smaller than 32, but still achieve higher accuracy than training an unpruned model of hidden size 32 from scratch.

On the HT-1 and HT-5 datasets, pruning the largest model still results in a speed-up factor of 1.3 (HT-1) and 1.5 (HT-5) compared to the unpruned model. This means that on these datasets, the pruned large models do not run faster than the smaller models, but are still significantly more accurate.

On all devices, previously published ESR baselines [20], [25] can be met by models pruned to considerable sparsity. In [25], listening tests were performed, in which the sound quality of the best-performing LSTM models of the HT-5 and Mesa amplifiers was rated as "excellent".

The rest of this section explores different aspects of the proposed method in more detail.

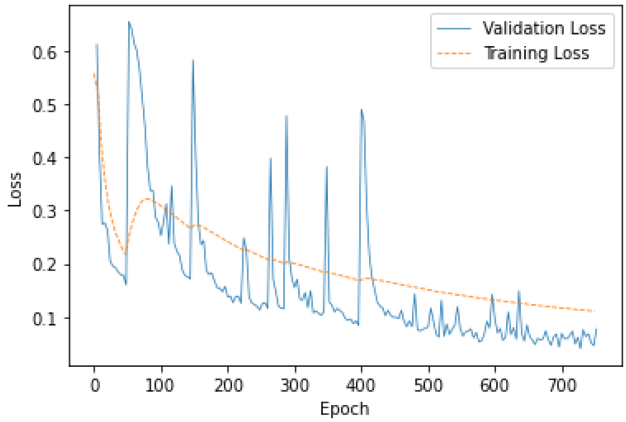

Fig. 2. Validation and training loss while training an LSTM model with hidden size 64 on the HT-5 dataset.

A. Mask Search Algorithm

Preliminary experiments showed that the FTF variant of the mask search algorithm, in which the unpruned model is trained to convergence before applying the EB early stopping criterion, consistently produced more accurate models at every stage of pruning than the variant in which the EB early stopping is applied in every iteration. One possible reason for this can be found in the relatively slow convergence of LSTM models when compared to the convolutional models, for which the EB criterion was first proposed [45]. Fig. 2 shows that the validation loss shows significant improvements until very late into training, suggesting that the important connectivity patterns of LSTM models take longer to emerge than those of convolutional models.

It is impractical to fully retrain the model at every mask search iteration to select the best-performing one, or to know up to which degree of sparsity a model can be pruned without major performance losses. With the FTF variant of the algorithm however, the validation loss of the model at the end of a mask search serves as a reasonable heuristic to predict the point at which pruning degrades the performance after retraining. Fig. 3 shows an example of the relationship between the validation loss at the end of a mask search iteration and the test loss achieved after fully retraining the model.

Figs. 4 and 5 compare the outputs of a model with hidden size 96. In one case, it was pruned to a sparsity of 91.8% with the

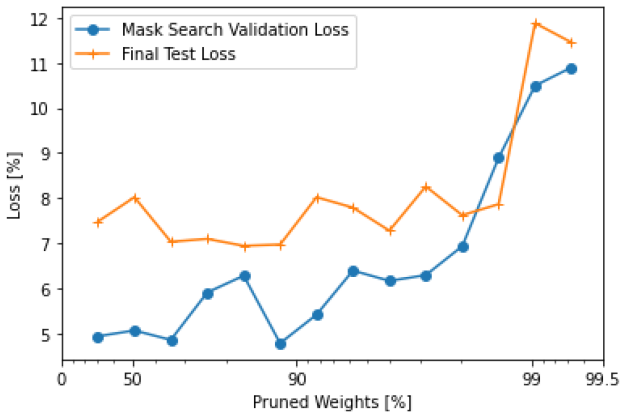

Fig. 3. The relationship between the validation loss at the point of early stopping and the final test loss while pruning an LSTM model with hidden size 32 and training on the Big Muff dataset.

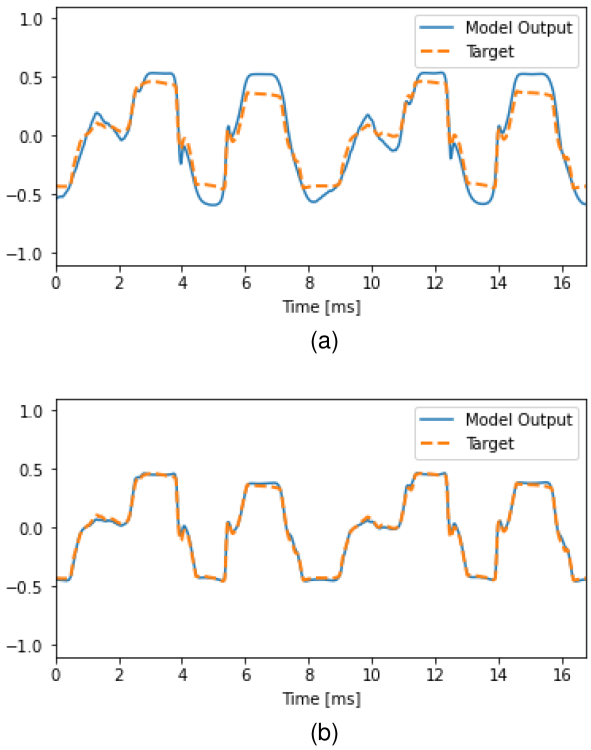

Fig. 4. A waveform processed through the HT-1 amplifier (target) and an LSTM model. (a) Output of the model pruned with one-shot pruning. (b) Output of the model pruned with iterative mask search.

iterative mask search algorithm, achieving an ESR test loss of 1.2% after retraining for 750 epochs. In the other, it was pruned to 90% sparsity in one shot, achieving an ESR test loss of 9.6% after retraining for 750 epochs. Across all datasets and model sizes, one-shot pruning results in a significant drop of the model accuracy at sparsities exceeding 50%, while the mask search algorithm is able to retain the model accuracy with sparsities up to 99.6%, depending on the model size and dataset.

Fig. 4 shows the outputs as a time-domain waveform, while Fig. 5 shows the frequency-domain spectra. Fig. 5 illustrates that even the model pruned in one shot matches the low-frequency content reasonably well, but starts to deviate from the target peaks at frequencies higher than ca. 2400 Hz. The model pruned using iterative mask search never misses the target peaks by more than 2 dB at all frequency ranges. The model pruned in one shot, however, misses most target peaks at higher frequencies by 3 dB or more.

B. Difference Between Devices

We were able to remove a significantly higher number of hidden nodes and thus achieve a larger speedup on the Mesa and Big Muff datasets than on the HT-1 and HT-5 datasets. Fig. 6 shows the differences between the datasets. This discrepancy does not seem to be related to the model accuracy, since the models were able to emulate the Mesa and HT-1 amplifiers more accurately than the Big Muff pedal and the HT-5 amplifier.

Fig. 7 shows the settling time of the models of the different devices, which is the time it takes to return to a steady state after excitation with an impulse. The HT-1 and HT-5 amplifier models, which proved harder to prune, show a significantly longer settling time. The need to model longer time dependencies is a likely reason for why these models require a larger number of effective hidden nodes and weights, than models of devices with shorter settling times.

Since the amount of pruning that can be achieved seems to be related to the effect's settling time, it is not possible to settle on a "one-size-fits-all" level of pruning. However, as described in Section V-A and Fig. 3, monitoring the validation loss at every pruning indication gives a good indication on when pruning starts to degenerate model performance, so that an appropriately pruned model may be selected.

C. Real-Time Performance

The removal of hidden nodes and the resulting size reduction of the matrix multiplications required for inference has a more significant impact on the inference speed of the models than the amount of pruned weights. It can be seen in Table I that a much higher speedup is achieved in the pruned models compared to their baselines when many hidden nodes can be removed.

At the relatively small hidden sizes of the models we are considering, sparsity alone does not benefit inference speed much. Fig. 8 shows that the sparse matrix type provided by the Eigen library incurs an initial significant overhead, so that a speed-up compared to an unpruned baseline using a dense implementation is only achieved once the sparsity exceeds 90%. However, as we are able to remove more and more nodes at higher sparsities, the implementation using dense matrix types still outperforms the sparse implementation.

D. Listening Test

To investigate the impact of pruning on the perceptual quality of the models, a MUSHRA-style listening test was carried out for the HT-1 amplifier and Big Muff pedal. For the HT-1 Amplifier, the unpruned model with hidden size 96 was compared to the same model after pruning, with a new hidden size 76. These models achieved a test loss of 1.0 and 1.2 respectively, indicating

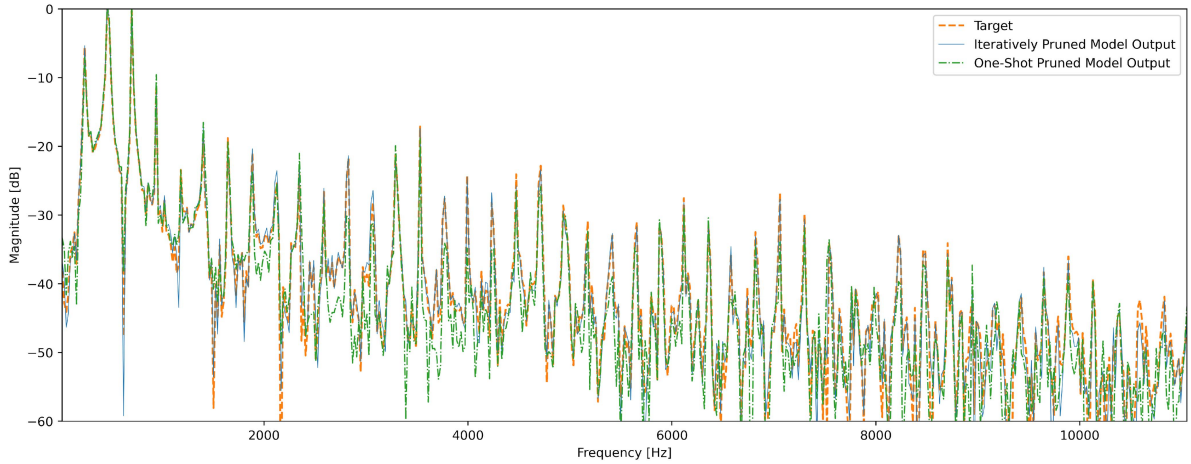

Fig. 5. The spectra of the outputs of the HT-1 amplifier (target) and of an LSTM model on the same input compared at different frequency ranges. The one-shot pruned model starts to deviate from the target significantly at around 2 kHz.

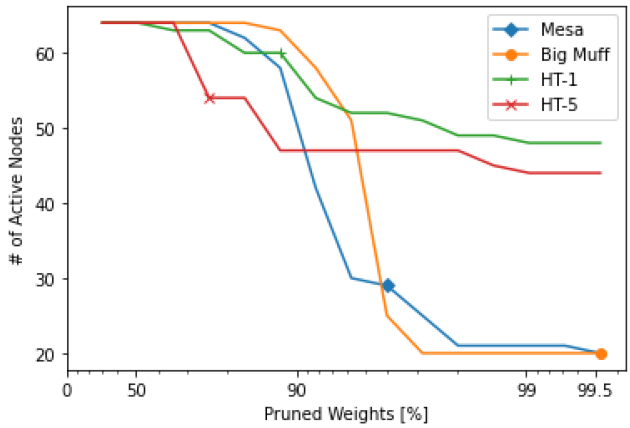

Fig. 6. The relationship between pruning and the number of active hidden nodes illustrated for an LSTM with hidden size 64. The models shown in table I are marked along the graph.

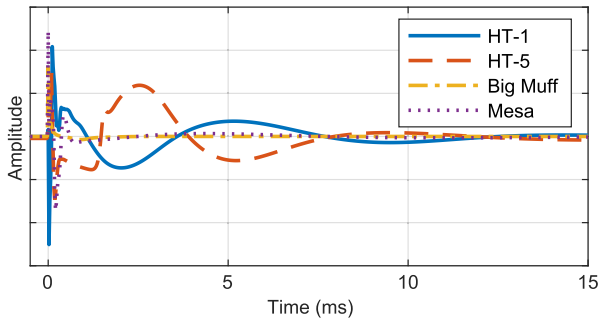

Fig. 7. Illustration of the settling times for unpruned LSTM models with hidden size 64 for each of the devices.

a slight decrease in quality from the pruning. For the Big Muff pedal, the unpruned model of hidden size 32 was compared to the model with initial hidden size of 64, after pruning to a hidden size of 20. In this case, the test loss for the two models was 7.3 and 5.4 respectively, indicating that the smaller pruned model performs better than the unpruned model.

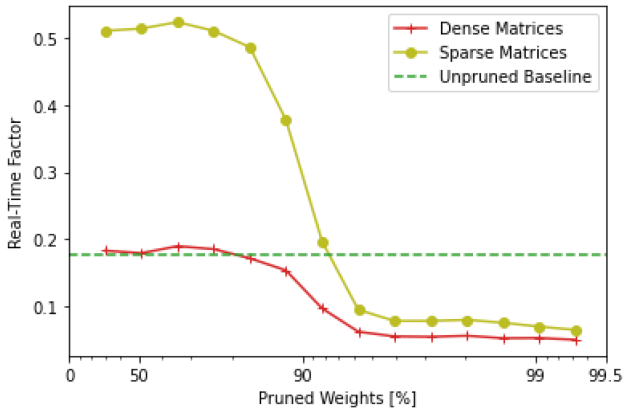

Fig. 8. Comparison of the real-time performance of a deep neural network model with hidden size 64, trained on the Big Muff dataset, using dense and sparse matrix types. Smaller is better.

For each trial, the participant was presented with a reference, consisting of a short clip of either guitar or bass guitar, that was processed through the target device. They were then asked to rate a number of test conditions based on perceived similarity to the reference, on a scale of 0 to 100. The test conditions included the hidden reference, a low anchor produced by processing the input through a $tanh$ nonlinearity, as well as the output of pruned and unpruned neural network models. This resulted in a total of 4 test conditions per trial. For each tested device, five examples from a bass guitar and two examples from an electric guitar were presented. More examples from the bass guitar were included as preliminary listening tests indicated that bass guitar produces more audible differences between the RNN models and the reference. This resulted in a total of seven MUSHRA trials for each device. In total, twelve people participated in the listening tests, with one being removed in post-screening because they gave the hidden reference a score of less than 90 in more than 15% of the trials. None of the participants reported hearing impairments. The tests were conducted in sound-proof booths using Sennheiser HD-650 headphones.

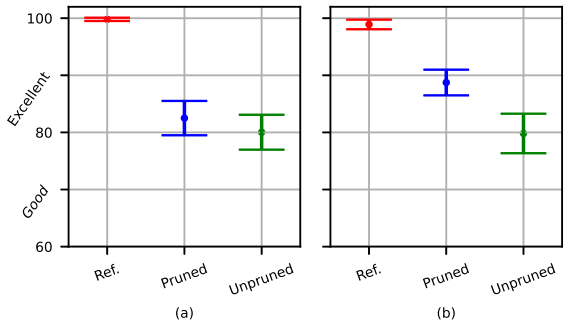

Fig. 9. Mean MUSHRA scores and 95% confidence intervals for pruned and unpruned models of the (a) HT-1 and (b) Big Muff, showing that pruning does not compromise the perceived sound quality but may rather improve it.

The results are shown in Fig. 9, where it can be seen that for the HT-1 the pruning has not resulted in a significant change in the perceptual quality of the model. In this case, the RTF of the unpruned and pruned models is 0.246 and 0.187, respectively, showing that the pruning has reduced the RTF by almost 25% whilst leaving the emulation quality unaffected. For the Big Muff, the unpruned and pruned models have RTFs of 0.091 and 0.057 respectively, a reduction of 37%. In this case, the results indicate that the pruned model offers a small, but statistically significant improvement over the unpruned model, showing that the pruning in this case has produced a model that both has a lower computational cost at runtime, and a higher perceived quality.

VI. Conclusion

This work investigated a new application of recent pruning methods to speed up the inference of neural network models of distortion effects. By employing iterative magnitude pruning, we were able to sparsify the deep neural networks to a high degree without a loss of accuracy. We applied the pruning to LSTM models of the Big Muff distortion pedal, the Blackstar HT-1 and HT-5 amplifiers, and the Mesa Boogie 5:50 Express Plus amplifier.

Speedups and significant parameter reductions were achieved on all four devices. The weights could be pruned by 65.7 to 99.6% in the different models. Training a large deep neural network model and pruning it consistently outperformed the unpruned model, according to both objective error metrics and a perceptual listening test. However, the Mesa and Big Muff models could be sparsified to a notably larger degree than the other two models, allowing the removal of a large number of hidden nodes, which is the main driver of faster inference. There exists a negative correlation between the settling time of an distortion device and the degree of achieved pruning, implying that our approach helps uncover the amount of hidden nodes required for the model to effectively capture the temporal behavior of the effect.

Acknowledgment

The authors acknowledge the computational resources provided by the Aalto Science-IT project.

References

[1] V. Välimäki, S. Bilbao, J. O. Smith, J. S. Abel, J. Pakarinen, and D. Berners, "Virtual analog effects," in *DAFX: Digital Audio Effects: 2nd Edition*, U. Zölzer, Ed. 2011, pp. 473–522.

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

[3] S. D'Angelo, "Lightweight virtual analog modeling," in *Proc. 22nd Colloq. Music Informat.*, Udine, Italy, 2018, pp. 59–63.

[4] 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. Process.*, vol. 18, no. 4, pp. 728–737, May 2010.

[5] D. T. Yeh, "Automated physical modeling of nonlinear audio circuits for real-time audio effects–part II: BJT and vacuum tube examples," *IEEE Trans. Audio Speech Lang. Process.*, vol. 20, no. 4, pp. 1207–1216, May 2012.

[6] J. Pakarinen and M. Karjalainen, "Enhanced wave digital triode model for real-time tube amplifier emulation," *IEEE Trans. Audio Speech Lang. Process.*, vol. 18, no. 4, pp. 738–746, May 2010.

[7] R. Giampiccolo, A. Bernardini, G. Gruosso, P. Maffezzoni, and A. Sarti, "Multiphysics modeling of audio circuits with nonlinear transformers," *J. Audio Eng. Soc.*, vol. 69, no. 6, pp. 374–388, Jun. 2021.

[8] B. C. Holmes, "Guitar effects-pedal emulation and identification," Ph.D. dissertation, Queen's Univ. Belfast, Fac. Eng. Phys. Sci., Belfast, Northern Ireland, U.K., Jul. 2019.

[9] F. Esqueda, B. Kuznetsov, and J. D. Parker, "Differentiable white-box virtual analog modeling," in *Proc. IEEE 24th Int. Conf. Digit. Audio Effects*, Vienna, Austria, 2021, pp. 41–48.

[10] M. J. Kemp, "Analysis and simulation of non-linear audio processes using finite impulse responses derived at multiple impulse amplitudes," in *Proc. 106th Audio Eng. Soc. Conv.*, Munich, Germany, 1999, p. 4919.

[11] T. Hélie, "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.*, vol. 18, no. 4, pp. 747–759, May 2010.

[12] S. Orcioni, A. Terenzi, S. Cecchi, F. Piazza, and A. Carini, "Identification of Volterra models of tube audio devices using multiple-variance method," *J. Audio Eng. Soc.*, vol. 66, no. 10, pp. 823–838, Oct. 2018.

[13] F. Eichas and U. Zölzer, "Black-box modeling of distortion circuits with block-oriented models," in *Proc. 19th Int. Conf. Digit. Audio Effects*, Brno, Czech Republic, 2016, pp. 5–9.

[14] J. Schattschneider and U. Zölzer, "Discrete-time models for nonlinear audio systems," in *Proc. 2nd COST G-6 Workshop Digit. Audio Effects*, Trondheim, Norway, 1999, pp. 45–48.

[15] D. J. Gillespie and D. P. Ellis, "Modeling nonlinear circuits with linearized dynamical models via kernel regression," in *Proc. IEEE Workshop Appl. Signal Process. Audio Acoust.*, New Paltz, NY, 2013, pp. 1–4.

[16] F. Eichas and U. Zölzer, "Gray-box modeling of guitar amplifiers," *J. Audio Eng. Soc.*, vol. 66, no. 12, pp. 1006–1015, Dec. 2018.

[17] J. D. Parker, F. Esqueda, and A. Bergner, "Modelling of nonlinear state-space systems using a deep neural network," in *Proc. 22nd Int. Conf. Digit. Audio Effects*, Birmingham, U.K., 2019, pp. 2–6.

[18] A. Wright and V. Välimäki, "Neural modeling of phaser and flanging effects," *J. Audio Eng. Soc.*, vol. 69, no. 7/8, pp. 517–529, Jul./Aug. 2021.

[19] Z. Zhang, E. Olbrych, J. Bruchalski, T. J. McCormick, and D. L. Livingston, "A vacuum-tube guitar amplifier model using long/short-term memory networks," in *Proc. IEEE SOUTHEASTCON*, 2018, pp. 1–5.

[20] A. Wright, E. P. Damskägg, and V. Välimäki, "Real-time black-box modelling with recurrent neural networks," in *Proc. 22nd Int. Conf. Digit. Audio Effects*, Birmingham, U.K., 2019, pp. 173–180.

[21] M. A. M. Ramírez, E. Benetos, and J. D. Reiss, "Deep learning for black-box modeling of audio effects," *Appl. Sci.*, vol. 10, no. 2, 2020, Art. no. 638.

[22] E. P. Damskägg, L. Juvela, E. Thuillier, and V. Välimäki, "Deep learning for tube amplifier emulation," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, Brighton, U.K., 2019, pp. 471–475.

[23] A. V. D. Oord et al., "WaveNet: A generative model for raw audio," 2016, *arXiv:1609.03499*.

[24] E. P. Damskägg, L. Juvela, and V. Välimäki, "Real-time modeling of audio distortion circuits with deep learning," in *Proc. Sound Music Comput. Conf.*, Malaga, Spain, 2019, pp. 332–339.

[25] A. Wright, E. P. Damskägg, L. Juvela, and V. Välimäki, "Real-time guitar amplifier emulation with deep learning," *Appl. Sci.*, vol. 10, no. 3, 2020, Art. no. 766.

[26] B. Kuznetsov, J. D. Parker, and F. Esqueda, "Differentiable IIR filters for machine learning applications," in *Proc. 23rd Int. Conf. Digit. Audio Effects*, Vienna, Austria, 2020, pp. 297–303.
[27] S. Nercessian, A. Sarroff, and K. J. Werner, "Lightweight and interpretable neural modeling of an audio distortion effect using hyperconditioned differentiable biquads," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, Toronto, Canada, 2021, pp. 890–894.
[28] C. Steinmetz and J. D. Reiss, "Efficient neural networks for real-time modeling of analog dynamic range compression," in *Proc. 152nd Audio Eng. Soc. Conv.*, 2022.
[29] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," *Adv. Neural Inf. Process. Syst.*, pp. 598–605, 1990.
[30] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," in *Proc. 29th Conf. Neural Info. Process. Syst.*, Montréal, Canada, 2015, pp. 1135–1143.
[31] J. H. Luo, J. Wu, and W. Lin, "ThiNet: A filter level pruning method for deep neural network compression," in *Proc. IEEE Int. Conf. Comput. Vis.*, Venice, Italy, 2017, pp. 5068–5076.
[32] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient ConvNets," in *Proc. 5th Int. Conf. Learn. Representations*, Toulon, France, 2017.
[33] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," in *Proc. 5th Int. Conf. Learn. Representations*, Toulon, France, 2017.
[34] S. Narang, E. Elsen, G. Diamos, and S. Sengupta, "Exploring sparsity in recurrent neural networks," in *Proc. 5th Int. Conf. Learn. Representations*, Toulon, France, 2017.
[35] W. Wen et al., "Learning intrinsic sparse structures within long short-term memory," in *Proc. 6th Int. Conf. Learn. Representations*, Vancouver, Canada, 2018.
[36] S. Liu, D. Constantin, M. Yulong, and P. Mykola, "Selfish sparse RNN training," in *Proc. 38th Int. Conf. Mach. Learn.*, 2021, pp. 6893–6904.
[37] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *Proc. 7th Int. Conf. Learn. Representations*, New Orleans, LA, 2019.
[38] P. Esling, N. Devis, A. Bitton, A. Caillon, A. Chemla-Romeu-Santos, and C. Douwes, "Diet deep generative audio models with structured lottery," in *Proc. 23rd Int. Conf. Digit. Audio Effects*, Vienna, Austria, 2020, pp. 317–324.
[39] J. H. Engel, L. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable digital signal processing," in *Proc. 8th Int. Conf. Learn. Representations*, 2020.
[40] A. Wright and V. Välimäki, "Perceptual loss function for neural modeling of audio systems," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, Barcelona, Spain, 2020, pp. 251–255.
[41] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.
[42] A. Renda, J. Frankle, and M. Carbin, "Comparing rewinding and fine-tuning in neural network pruning," in *Proc. 8th Int. Conf. Learn. Representations*, Addis Ababa, Ethiopia, 2020.
[43] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, "Rethinking the value of network pruning," in *Proc. 7th Int. Conf. Learn. Representations*, New Orleans, LA, 2019.
[44] A. Achille, M. Rovere, and S. Soatto, "Critical learning periods in deep neural networks," in *Proc. 7th Int. Conf. Learn. Representations*, New Orleans, LA, 2019.
[45] H. You et al., "Drawing early-bird tickets: Towards more efficient training of deep networks," in *Proc. 8th Int. Conf. Learn. Representations*, Addis Ababa, Ethiopia, 2020.
[46] C. Kehling, J. Abeßer, C. Dittmar, and G. Schuller, "Automatic tablature transcription of electric guitar recordings by estimation of score- and instrument-related parameters," in *Proc. 17th Int. Conf. Digit. Audio Effects*, Erlangen, Germany, 2014, pp. 219–226.
[47] J. Abeßer, P. Kramer, C. Dittmar, and G. Schuller, "Parametric audio coding of bass guitar recordings using a tuned physical modeling algorithm," in *Proc. 16th Int. Conf. Digit. Audio Effects*, Maynooth, Ireland, 2013, pp. 2–5.
[48] T. Schmitz and J. J. Embrechts, "Introducing a dataset of guitar amplifier sounds for nonlinear emulation benchmarking," in *Proc. Audio Eng. Soc. 145th Conv.*, New York, NY, 2018.
[49] K. J. Werner, V. Nangia, J. O. Smith, and J. S. Abel, "Resolving wave digital filters with multiple/multiport nonlinearities," in *Proc. 18th Int. Conf. Digit. Audio Effects*, Trondheim, Norway, 2015, pp. 31–33.
[50] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Representations*, San Diego, CA, 2015.
[51] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. 32nd Conf. Neural Info. Process. Syst.*, Vancouver, BC, Canada, 2019, pp. 8024–8035.
[52] W. Falcon and the PyTorch lightning team, "PyTorch lightning," Mar. 2019. [Online]. Available: https://github.com/PyTorchLightning/pytorch-lightning
[53] P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proc. IEEE*, vol. 78, no. 10, pp. 1550–1560, Oct. 1990.
[54] J. L. Elman, "Finding structure in time," *Cogn. Sci.*, vol. 14, no. 2, pp. 179–211, Mar. 1990.

David Südholt received the B.Sc. degree in scientific programming from the FH Aachen University of Applied Sciences, Aachen, Germany, in 2016. He is currently working toward the M.Sc. degree in sound and music computing with Aalborg University Copenhagen, Denmark. His research interests include physical modeling and deep learning for audio systems.

Alec Wright received the M.Eng. degree in mechanical engineering from The University of Manchester, Manchester, U.K., in 2014, and the M.Sc. degree in acoustics and music technology from the University of Edinburgh, Edinburgh, U.K., in 2018. He is currently working toward the Doctoral degree with the Acoustics Lab of Aalto University, Espoo, Finland. His research interests include virtual analog modelling of guitar amplifiers and other audio effects and machine learning for audio applications.

Cumhur Erkut (received the M.Sc. and D.Sc. degrees in 1997 and 2002, respectively, and the Ph.D. degree in acoustics and audio signal processing from the Helsinki University of Technology, Finland, with minor studies on Information Systems (Machine Learning). He is currently an Associate Professor of sonic and embodied interaction with Aalborg University Copenhagen. Dr. Erkut is an Associate Editor for Frontiers in Audio Signal Processing, and is with the steering committee of International Conference on Movement Computing (MOCO).

Vesa Välimäki (Fellow, IEEE) received the M.Sc. and D.Sc. degrees in electrical engineering from the Helsinki University of Technology (TKK), Espoo, Finland, in 1992 and 1995, respectively. He was a Postdoctoral Research Fellow with the University of Westminster, London, U.K., in 1996. In 1997–2001, he was a Senior Assistant (cf. Assistant Professor) with TKK. In 2001–2002, he was a Professor of signal processing with the Pori unit, Tampere University of Technology, Pori, Finland. In 2006–2007, he was the Head of the TKK Laboratory of Acoustics and Audio Signal Processing. In 2008–2009, he was a Visiting Scholar with Stanford University, Stanford, CA, USA. He is currently a Full Professor of audio signal processing and the Vice Dean for Research in electrical engineering with Aalto University, Espoo, Finland. His research interests include audio and musical applications of signal processing and machine learning. Prof. Välimäki is a Fellow of the Audio Engineering Society, and a Life Member of the Acoustical Society of Finland. In 2007–2013 he was a Member of the Audio and Acoustic Signal Processing Technical Committee of the IEEE Signal Processing Society and is currently an Associate Member. In 2005–2009, he was an Associate Editor of the IEEE SIGNAL PROCESSING LETTERS and in 2007–2011, as an Associate Editor for the IEEE TRANSACTIONS ON AUDIO, SPEECH AND LANGUAGE PROCESSING. In 2015–2020, he was a Senior Area Editor of the IEEE/ACM TRANSACTIONS ON AUDIO, SPEECH AND LANGUAGE PROCESSING. He was with the Editorial Board of the *Research Letters in Signal Processing* and of the *Journal of Electrical and Computer Engineering*. In 2007, 2015, and 2019, he was a Guest Editor of special issues of the *IEEE Signal Processing Magazine*, and in 2010, of a special issue of the IEEE TRANSACTIONS ON AUDIO, SPEECH AND LANGUAGE PROCESSING. In 2016, 2018, and 2020, he was a Guest Editor of special issues published in *Applied Sciences*. He was the Chair of the 2008 International Conference on Digital Audio Effects (DAFX) and the Chair of the 2017 International Conference on Sound and Music Computing. He is currently the Editor-in-Chief of the *Journal of the Audio Engineering Society*.