EURASIP Journal on Audio, Speech, and Music Processing

EMPIRICAL RESEARCH

Open Access

Blind extraction of guitar effects through blind system inversion and neural guitar effect modeling

Reemt Hinrichs[1*], Kevin Gerkens[1], Alexander Lange[1] and Jörn Ostermann[1]

Abstract

Audio effects are an ubiquitous tool in music production due to the interesting ways in which they can shape the sound of music. Guitar effects, the subset of all audio effects focusing on guitar signals, are commonly used in popular music to shape the guitar sound to fit specific genres or to create more variety within musical compositions. Automatic extraction of guitar effects and their parameter settings, with the aim to copy a target guitar sound, has been previously investigated, where artificial neural networks first determine the effect class of a reference signal and subsequently the parameter settings. These approaches require a corresponding guitar effect implementation to be available. In general, for very close sound matching, additional research regarding effect implementations is necessary. In this work, we present a different approach to circumvent these issues. We propose blind extraction of guitar effects through a combination of blind system inversion and neural guitar effect modeling. That way, an immediately usable, blind copy of the target guitar effect is obtained. The proposed method is tested with the phaser, softclipping and slapback delay effect. Listening tests with eight subjects indicate excellent quality of the blind copies, i.e., little to no difference to the reference guitar effect.

Keywords Demucs, Neural effect modeling, Guitar effect extraction, Blind system identification

1 Introduction

Audio effects are a common tool used in the production of music [1]. They find use in all sorts of music and are applied to all kinds of instruments such as guitar, keyboard, and vocals [2]. In guitar-centered music, a prominent and well known effect is the distortion effect, closely related to the overdrive effect. A multitude of other effects exist such as phaser, delay, and ring-modulator [3]. Music genres like Post-Rock, achieve part of their distinctive sound due to the usage of certain audio or guitar effects. A common question is how a certain instrument or guitar sound was created. The aim usually is to emulate that sound, or to use it as basis for ones own sound. Automatic replication of guitar effects that yield a desired target sound can be of interest. For this purpose, algorithms are required that allow to copy a given target reference guitar sound.

Many common guitar effects belong to well defined classes [4], e.g., overdrive, echo, or reverb among others. Additionally, their sound can usually be modified by one or more parameters. Also, these effects can be and often are combined in an effect chain. Previous work in this area, loosely termed guitar effect extraction, is based on the classification and parameter extraction of the effects [5, 6]. The overarching goal is to allow automatic replication of a target guitar sound using another guitar. This work is a continuation of our previous works [6–8] and attempts to remedy some of the observed shortcomings. As such, emphasis in the background section is given to methods based on or closely related to effect and

*Correspondence:
Reemt Hinrichs
hinrichs@tnt.uni-hannover.de
[1] Institut für Informationsverarbeitung, Leibniz University Hannover, Hannover, Germany

parameter setting extraction. In the following section, previous related work is summarized before explaining the idea of this manuscript.

1.1 Background

Early work about guitar effect extraction was concerned with the classification of guitar effects. Stein et al. pioneered this area of research with their seminal work [5, 9], using a large set of audio features and a support vector machine to classify eleven guitar effects. They achieved an accuracy of 97.7 % for solo guitar recordings. Further work regarding the classification of guitar effects was done by Eichas et al. [10] and Schmitt et al. [11], the latter investigating the importance of audio features and comparing the so called bag-of-audio-words approach to the use of functionals, i.e., feature statistics like moments of order *k*. They found both approaches to achieve similar high performance.

The issue of deriving or extracting the parameter settings of a guitar effect from audio samples is closely related to sound matching [12], the estimation of synthesizer parameters replicating a certain given sound. Yee-King et al. [13] proposed a special long short-term memory network structure for this purpose, and achieved a close sound for 25 % of their test set.

Sheng and Fazekas [14] investigated extraction of dynamic range compression parameter settings from audio samples of violins and drums using several audio features and regression models. For the same purpose, the authors investigated deep neural networks [15] and found them to improve performance, predicting multiple dynamic range compression parameters at once from monophonic and polyphonic audio samples.

Research regarding the extraction of guitar effect parameter settings is scarce. So far, only four previous works exist: Jürgens et al. [6] pioneered this task using shallow neural networks combined with specifically designed features for each guitar effect, achieving or surpassing the (presumed) performance of a human expert. Comunitá et al. [16] used convolutional neural networks (CNNs) to extract the parameter settings of different implementations of distortion related guitar effects from monophonic and polyphonic audio samples, achieving below 0.1 root mean-square error in all cases. In [7, 8], a CNN was used for classification of guitar effects from instrument mixes as well as extraction of their parameter settings. The CNN was used for the extraction of single and multi guitar effects limited to distortion, tremolo, and slapback delay effect. The CNN yielded presumed human expert level results for all effects considered but saw some increase in parameter extraction error when extracting the settings from multi-effect samples. The most recent publication closely related to the aforementioned works is Lee et al. [17]. They investigated blind estimation of audio processing graphs, which is a generalization to the general problem of deriving a sequence of processing steps, including parameter settings, that were applied to a clean audio signal from target, processed audio. Unlike the aforementioned publications, they are not specifically concerned with guitar effects and are using transformers instead of CNNs. While they were able to achieve reasonable to good quality, they saw a considerable drop in performance on unseen data. A similar approach concerned with blind audio effect estimation based on autoencoders was recently proposed by Peladeau and Peeters [18]. They estimated effect type and parameter settings for equalizer, compressor, and clipper and found that performance was best, when optimizing the autoencoder with respect to a perceptual loss instead of the parameter extraction error.

The publication most similar to ours is concerned with blind adversarial guitar amplifier modeling [19]. Wright et al. attempt to copy a target timbre of a guitar, including possible distortion effects, to another guitar using generative neural networks. Their approach does not extract parameter settings or effect classes but directly replicates a target sound. In their MUSHRA evaluation, they achieved a perceptual match between Good and Excellent on the MUSHRA scale for their copied guitar timbres. A comparison of our work to the work of Wright et al. can be found in Sec. 4.

Except for Wright et al., the previously described approaches in guitar effect extraction have the minor downside of requiring hard- or software implementations of the recognized guitar effects to actually make use of the extracted effect types and parameter settings. More specifically, very close or exact replication of a target guitar sound in general requires additional research/knowledge regarding implementation types, as in general different implementations of guitar effects at least slightly differ from each other in the produced sound.

Furthermore, when extracting parameter settings from guitar samples processed by multi-effects, i.e., a sequence or chain of effects, as in [8], CNNs specifically trained on multi-effect samples were used. Hinrichs et al. [8] observed an increase of about 50% in parameter extraction error when using CNNs, trained on single effect samples, for the extraction of parameter settings of multi-effect samples. The authors conclude, that CNNs specifically trained on multi-effects are required to achieve high performance on multi-effect processed samples. Due to the general nonlinear nature of many guitar effects, the order of effects matters, and as such

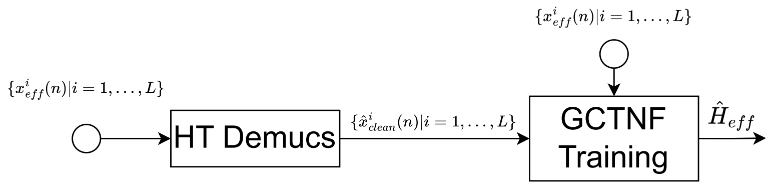

$\{x^i_{eff}(n)|i=1,\dots,L\}$
HT Demucs
$\{\hat{x}^i_{clean}(n)|i=1,\dots,L\}$
$\{x^i_{eff}(n)|i=1,\dots,L\}$
GCTNF Training
$\hat{H}_{eff}$

Fig. 1 Depiction of the proposed idea: a previously, and separately, trained Hybrid Transformer Demucs (HT Demucs) network estimates for a set of processed reference input guitar signals $x^i_{eff}(n)$, each processed by the same effect H_{eff}, the clean, unprocessed reference signal $\hat{x}^i_{clean}$. These estimates are then used, together with the given processed signals $x^i_{eff}(n)$, to train a gated convolutional network with temporal feature-wise linear modulation (GCNTF). The result is an estimate $\hat{H}_{eff}$ of the actual guitar effect H_{eff} used to process the x^i_{clean}, which then can be applied to any guitar signal

training specific CNNs for a larger number of guitar effects is not feasible.

1.2 Contribution

In this work, a novel approach to guitar effect extraction is investigated, which, in principle, can remedy all or most described downsides: for a given reference guitar signal, which has been processed by some guitar effect(s), the original, clean guitar signal, i.e., the reference guitar signal before being processed by said guitar effect, is estimated or regenerated using an artificial neural network, namely Hybrid Transformer Demucs (HT Demucs). HT Demucs achieves state-of-the-art results for declipping of guitar signals [20] and, more generally, audio effect removal [21] and thus appears to be a natural choice for the task of guitar effect removal. Then, using the pair of the regenerated, clean reference guitar signal and the processed reference guitar signal, the guitar effect is learned using another artificial neural network, namely gated convolutional network with temporal feature-wise linear modulation (GCNTF) proposed by Comunità et al. for neural guitar effect modeling [22]. After training, this second network then is a copy of the original guitar effect that was applied to the clean, reference guitar signal and can be used to modify arbitrary other guitar signals. The general signal flow of this approach is shown in Fig. 1. Thus, blind extraction, i.e., without knowledge of the original, clean input signal, of the guitar effect is achieved. No external effect implementations are required, and in principle, the target guitar sound can be approximated with arbitrary precision for any order of the applied guitar effects.

In this manuscript, we describe the precise process of blind guitar effect extraction in detail and evaluate our approach in subjects to provide evidence of its effectiveness given current technology. Furthermore, we discuss the current limitations of our approach.

2 Fundamentals

2.1 Problem description

Let x_{clean} be a clean reference guitar signal, subsequently processed by the linear or nonlinear guitar effect H_{eff}. The resulting output signal is given as

$$x_{eff} := H_{eff}\{x_{clean}\}. \tag{1}$$

In the first step of the proposed blind guitar extraction scheme, we estimate x_{clean}, i.e., we require some system G which approximately inverts H_{eff} to recover an approximation $\hat{x}_{clean}$ of x_{clean}. That is, we want to compute

$$\hat{x}_{clean} := G\{H_{eff}\{x_{clean}\}\} \tag{2}$$

such that

$$d(x_{clean}, \hat{x}_{clean}) \to min. \tag{3}$$

for some metric d of interest. This problem is a special kind of inverse problem or blind system identification. If such a system G is found, then, using $\hat{x}_{clean}$, we can approximately deduce H_{eff} using the input/output pair $(\hat{x}_{clean}, x_{eff})$ and another artificial neural network F, which is trained using the loss function $\mathcal{L}(y_F, x_{eff}) := d(y_F, x_{eff})$ with $y_F := F\{\hat{x}_{clean}\}$.

After training G and F, $d(F\{\hat{x}_{clean}\}, x_{eff})$ should be small and the output signal of F should sound quite similar to x_{eff}. F can now be used to modify other guitar signals, achieving blind extraction or copying of the target guitar effect. For simplicity of the discussion, here we assumed that x_{clean} is sufficiently long/rich in information or actually consists of a set of individual guitar samples allowing to train the network F. With recent advances [20], the system G, at least in certain cases, can be learned using HT Demucs. The system F can be learned using GCNTFs, which showed high performance in neural guitar effect modeling [22]. The limitations of this approach are discussed in Sec. 4.

2.1.1 Hybrid transformer demucs

Demucs [23] is a neural network architecture for music source separation, which consists of a U-Net like encoder/decoder architecture, each consisting of several convolutional blocks, and a bidirectional long-term short-term network (BiLSTM) between encoder and decoder. It separates a source audio signal using its immediate waveform. Hybrid Demucs [24] improves Demucs by introducing a second, separate U-Net like encoder/decoder architecture, which processes the audio signal's short-time Fourier transform. The two encoder/decoder structure are connected (and summed if dimensions allow it) through new shared convolutional block layers in the center of the architecture. Hybrid transformer Demucs [25] improves Hybrid Demucs by replacing the shared convolutional blocks with a cross-domain Transformer encoder. Hybrid transformer Demucs (HT Demucs) achieves state-of-the-art results in music source separation. Recently, its predecessor Demucs was found to achieve high performance in declipping of distorted guitar signals [20], i.e., it was used to recover clean guitar signals from distorted input signals. This is a special case of the problem of blind system inversion, which is a key part of the proposed idea.

Seeing that HT Demucs is a more powerful architecture than Demucs, at least for music source separation, we hoped that HT Demucs surpasses the performance of the original Demucs in Declipping and, more generally, blind system inversion, as well. This hope is supported by very recent results from [21], which found HT Demucs to be highly effective in general audio effect removal.

Initially, we also investigated gated recurrent units for effect removal from guitar signals; however, they did not yield sufficient quality of the regenerated clean signals, even for the most simple cases. In comparison, HT Demucs performed considerably better.

2.1.2 Gated convolution network with temporal FiLM

Neural modeling of guitar effects and amplifiers nowadays achieves high quality in most conditions. A good review of the field can be found in [26]. Recurrent neural networks are a general purpose tool used for neural effect modeling, where long short-term memory networks (LSTMs) and gated recurrent and convolutional networks are topologies commonly used [27]. Gated convolution networks with temporal FiLM (GCNTFs) combine gated convolutional networks (GCNs), a special kind of temporal convolutional network and the previous state-of-the-art in black box modeling of guitar effects, with temporal feature-wise linear modulation (temporal FiLM). The temporal FiLM is used to capture long range temporal dependencies in input signals and for the modeling of audio effects. In the latter case, it modulates the intermediate activations of the GCNs. That way, the GCNTF improves the state of the art at least for the modeling of fuzz and compressor [22]. In this work, the GCNTF is used as a tool for universal neural guitar effect modeling. In all our informal pilot tests it was found to yield very high quality replications of the tested guitar effects.

2.1.3 Configuration of HT Demucs and the GCNTF

The following changes were made to HT Demucs in this work compared to the default implementation [28]: we used only one audio channel, i.e., mono audio, instead of two, and 32 channels per layer instead of 48. We set the rewrite parameter, which, if set to true, adds additional convolutional layers, to false to reduce the computational complexity to a manageable level. For the encoder and decoder, 5 layers instead of 4 were used in order to slightly counteract the reduction in expressive power due to setting the rewrite parameter to false.

For the GCNTF, we used 10 layers, each consisting of 24 channels with kernel size and dilation growth set to 3. Initial investigations suggested that a longer receptive field yields superior performance for time-dependent guitar effects. The chosen setting yields a receptive field approximately equal to the length of the input signals. The total number of parameters of the GCNTF was 85,969. The receptive field covered about 59,000 samples.

2.2 Datasets and plugins

All training datasets in this work are based on the GuitarSet dataset [29]. It consists of 320 recordings by six guitarists with a total duration of about three hours. The recordings cover five different genres and several different tempos. The samples were initially split into segments of four seconds length each and downsampled to 16 kHz. Additionally, the dataset was augmented by creating five different versions of each sample through processing it with a 3-band equalizer using randomly sampled parameter settings. Parameter settings were sampled from [− 40 dB, − 0.1 dB]. The samples were subsequently peak normalized to a maximum absolute value of 1.0. This clean dataset contains 12,840 samples with a total duration of over 14 hours and is called GuitarsetEQ.

For each of the effects softclipping, phaser, and slapback delay, seven processed versions of each sample from the GuitarSetEQ dataset were created using random effect parameter settings. The parameter setting range was limited to commonly used values, namely [1 dB, 20 dB] for

Table 1 Overview of the investigated guitar effects, their parameters and the respective implementations used

Effect	Parameters	Source
Softclipping	Gain	Custom
Phaser	Rate, depth, mix	Pedalboard [30]
Slapback delay	Time, mix	Pedalboard [30]

Table 2 The parameter settings for each guitar effect used in the IDMTCFX dataset. The boldfaced settings were chosen for the MUSHRA listening test. For softclipping, the gain is given in dB

	Effects		
Setting number	Softclipping	Phaser	Slapback delay
1	**5**	**(0.3, 0.48, 0.43)**	**(0.15, 0.5)**
2	8	(0.3, 0.49, 0.4)	**(0.19, 0.22)**
3	**10**	**(1.0, 0.5, 0.5)**	(0.19, 0.42)
4	12	(1.37, 0.41, 0.36)	**(0.21, 0.44)**
5	**13**	**(1.81, 0.43, 0.38)**	(0.24, 0.43)

the gain setting of the softclipping effect, [0.3 Hz, 2.0 Hz] for the rate of the phaser effect, [0.05 s, 0.3 s] for the time parameter of the slapback delay, and [0.2, 0.5] for the other parameters of the latter two effects. The samples are peak normalized to a maximum absolute value of 1.0 after processing. This dataset is referred to as GuitarSetVFX and contains 89,880 samples per effect with a total duration of just under 100 h. The corresponding pairs of clean and processed samples from the GuitarSetEQ and GuitarSetVFX datasets are used to train a single HT Demucs network to learn effect removal. The application of phaser and slapback delay is done using Pedalboard [30], while softclipping is customarily implemented through the input/output relation

$$y = tanh(g \cdot x), \tag{4}$$

with the gain parameter g, the softclipping input signal x, and the output signal y. x is peak-normalized before applying softclipping. A summary of these effects is given in Table 1.

Another dataset called GuitarSetCFX was made by randomly sampling five different parameter setting tuples for each of the effects and creating a processed version of the GuitarSetEQ dataset for each tuple. This results in 15 sub-datasets with 12,840 samples and a duration of around 14 h each.

Lastly, a test dataset was created based on parts of the IDMT-SMT-GUITAR dataset. The recordings of its fourth sub-dataset, which were made with an Ibanez RG2820 guitar, were first split into segments of four seconds length each and downsampled to 16 kHz. This dataset, consisting of the unprocessed, clean samples, is referred to as IDMTClean. The samples were then processed with all effects and parameter settings given in Table 2. The resulting dataset is referred to as IDMTCFX.

2.3 Training

A single HT Demucs network was trained, covering all three investigated effects, using the GuitarSetEQ and GuitarSetVFX datasets for the target and source samples, respectively. A 80:20 training/validation split was used. Optimization was performed with the Adam optimizer and a learning rate of $3 \cdot 10^{-4}$. In total, 273 training epochs were performed, where the training was manually aborted once it had converged. For all networks, the loss function $\mathcal{L}$ was

$$\mathcal{L} = L1 + \text{MR-STFT}, \tag{5}$$

with the mean absolute error L1 and the multi-scale short-time Fourier transform (MR-STFT) loss [31], the latter taken from the auraloss python library [32]. By default, HT Demucs uses the signal-to-distortion ratio as its loss function. However, we found that the loss according to Eq. 5 yielded less artifacts and better overall quality.

The GCNTFs were trained, one for each effect, using the GuitarSetCFX dataset as target samples and the regenerated GuitarSetCFX, where the trained HT Demucs network regenerated the clean, unprocessed samples, used as source samples. The other training parameters were identical to the training of HT Demucs, except that 300 epochs were trained. All samples were peak-normalized before feeding them to the networks.

2.4 Listening test

To assess the quality of the copied guitar effects, we performed MUSHRA tests at the Institut für Informationsverarbeitung with a total of eight subjects, all of them in the age range of 18–27 years. All subjects had some level of musical experience, all playing an instrument. In total, three guitar effects were evaluated, namely softclipping, slapback delay, and phaser effect as summarized in Table 1. The order in which these guitar effects were presented was random for each subject. Furthermore, the samples corresponding to one guitar effect were presented in random order. The tests were performed double blind using the WebMUSHRA software [33]. The software shows a rating label for each intervals of 20 points lengths next to the rating controller that is supposed to guide the subject in its rating. These labels range from “bad,” for MUSHRA scores between zero and 20, up to “good,” for MUSHRA scores between 60

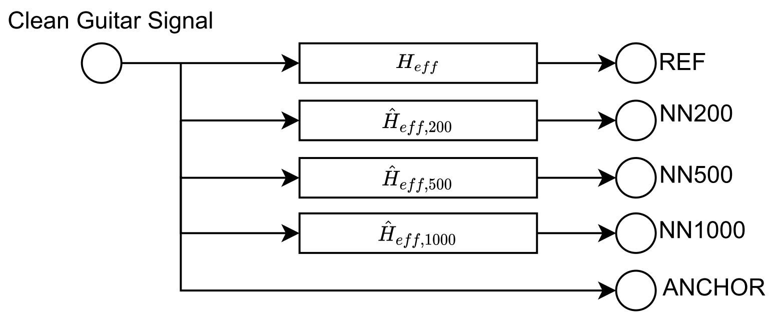


Fig. 2 Flowchart of the generation of the test stimuli of the MUSHRA test. H_{eff} is the respective reference guitar effect, e.g., Pedalboard phaser, $\hat{H}_{eff,K}$ is the learned guitar effect, obtained as depicted in Fig. 1, using K samples during training. By processing clean guitar signals by $\hat{H}_{eff,K}$ we obtain the **NN200**, **NN500** and **NN1000** conditions. The **ANCHOR** condition is the unprocessed clean guitar signal

and 80, and "excellent," for MUSHRA scores between 80 and 100. Before testing, all subjects were introduced into the general MUSHRA procedure by the main authors using a single example which was not part of the actual MUSHRA test. The subjects were additionally encouraged to rate the samples according to their own, subjective impression, emphasizing that any value ranging from zero to 100 was perfectly fine, if they considered it to be an accurate assessment, as long as it abides to the MUSHRA regulations. These regulations require one stimulus to be assigned the rating zero and one stimulus to be assigned the rating 100.

2.4.1 Conditions

For each of the three investigated effects, five test conditions were tested: the reference condition **REF**, which is the clean guitar signal processed by the reference guitar plugin, i.e., the reference effect. The anchor condition **ANCHOR**, which is the original clean guitar signal without any processing. Finally, the **NN200**, **NN500**, and **NN1000** conditions correspond to the original clean guitar samples processed by the GCNTFs after training with 200, 500, or 1000 training samples, respectively. The corresponding total durations of the training samples are 800 s, 2000 s, and 4000 s.

These training samples were chosen randomly and inclusively from the GuitarSetCFX dataset, i.e., such that all training samples of the **NN200** condition were part of the training samples of the **NN500** condition, and all training samples of the **NN500** condition were part of the training samples of the **NN1000** condition.

The reason for testing models trained using different numbers of samples was the fact that in a real scenario, where a target guitar sound of a song is supposed to be copied, at best about 3 min of guitar can be used. While still far away from a total duration of 3 min, investigating the impact of a reduced number of training samples was an interesting research question. Figure 2 depicts the signal flowchart for each condition. There, $H_{eff,K}$ is the GCNTF network after training with K samples.

2.4.2 Stimuli

For each considered guitar effect, three different parameter settings were considered, each applied to the same five clean guitar samples. The parameter settings were selected to depict a wide range of settings for at least one of the effect parameters. This means, for example, that for the phaser effect, one setting with slow, medium, and fast modulation rate each were chosen from the five tuples that were determined during the creation of the GuitarSetCFX dataset. The parameter settings used in the listening test are highlighted in Table 2 using bold font. Regarding the guitar samples, three of the five samples for each effect were chosen randomly, whereas the other two samples were chosen by hand because their respective musical styles were considered to fit the given effect. For softclipping, prior informal listening tests suggested that HT Demucs yields decreasing performance above a gain of about 15 db. This is why the highest considered gain is below 15 dB for softclipping. All samples used in the MUSHRA listening tests were loudness normalized according to the ITU-R BS.1770-4 recommendation [34].

The **ANCHOR** samples were taken from the IDMT-Clean dataset, with the **REF** samples being the corresponding processed samples from the IDMTCFX dataset. The stimuli were created by processing the clean guitar samples with the trained GCNTF models for the respective effect, parameter setting, and dataset size. Most importantly, for the softclipping effect only, each sample was processed twice by the GCNTF models since the effect was considered too weak after a single pass. The quality after a single pass was not per se bad but would have obviously been noticed by subjects of the listening tests. This is picked up on in the discussion. Some audio examples showcasing the effect removal of HT Demucs as well as a few samples from the MUSHRA listening test

Table 3 Overview of the effect implementations and parameters used for the GuitarSet VFX-3P data set

Effect	Producer	Parameter
Softclipping	Kilohearts Distortion Overdrive	Gain
	Kilohearts Distortion Hardclip	
	Kilohearts Distortion Saturate	
Phaser	BlueCat Audio	Rate, depth, mix
	Kilohearts	
	ChowDSP	Rate, depth
Slapback delay	FullBucket	Time, mix
	Kilohearts	
	HY	

Then, to assess generalization to unseen guitar or recording types, Output SDR and Input SDR was assessed for guitar samples taken from the IDMTCFX dataset. Finally, to assess the possible generalization capabilities to unseen effect implementations, a new dataset was created, hereonforth called GuitarSetVFX-3P, which took the samples of the GuitarSetEQ dataset, but now, for each effect and sample, one out of three effect implementations was randomly selected. These effect implementations are given in Table 3. Note that these were not used for the creation of the GuitarSetCFX dataset. The different softclipping implementations each used a different characteristic curve. HT Demucs was then newly trained on this GuitarSetVFX-3P dataset and evaluated with respect to Output-SDR and Input-SDR on the GuitarSetCFX dataset, which used (for this newly trained HT Demucs) unseen effect implementations. For each Output SDR and Input SDR assessment, 1000 randomly selected samples were used. Furthermore, using the results of the listening tests, we assessed the correlation of the MUSHRA Scores with the values of the L1, MR-STFT, and the actual loss $\mathcal{L} \equiv L1 + MR - STFT$, as given in Eq. 5, used for training the GCNTF and HT Demucs. This allows to give further insight into the importance of the different metrics on a per effect basis.

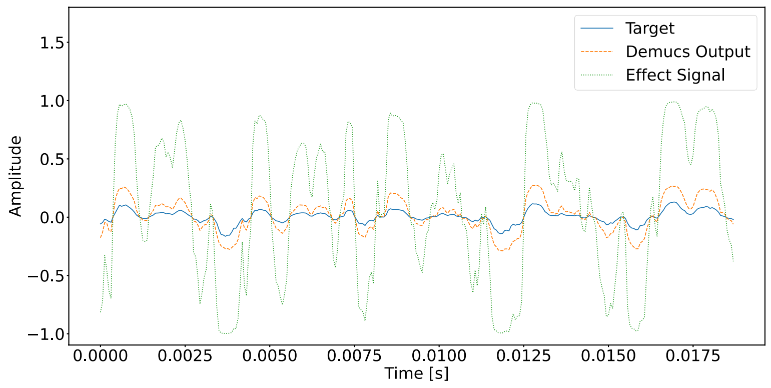


Fig. 3 Waveforms of a section of a clean guitar signal (Target), the same signal with a softclipping effect with 13 dB gain (Effect Signal), and the signal that was regenerated by Hybrid Transformer Demucs (Demucs Output)

can be found under https://rhtnt.github.io/BlindEffectExtraction/. The url furthermore contains a link to the complete set of stimuli of the MUSHRA test as well as additional audio material.

2.5 Objective evaluation

To assess some aspects of the proposed method objectively, we evaluated the samples regenerated by HT Demucs with respect to signal-to-distortion ratio (SDR) by comparing the regenerated samples to the corresponding original, unprocessed clean audio samples. This SDR, hereonforth called Output SDR, was compared to the SDR of the processed samples before application of HT Demucs, hereonforth called Input SDR. If HT Demucs is effective in removing the guitar effects, the difference between the Output SDR and the Input SDR should be positive. This objective evaluation was performed on three different datasets/test cases: first, the Output SDR and Input SDR was assessed for guitar samples coming from the validation split of the dataset used for training HT Demucs, i.e., the GuitarSetVFX dataset.

3 Results

Figure 3 shows an example of a regenerated waveform created by HT Demucs, where the softclipping effect is approximately removed. Some mild artifacts remain, but the original waveform is approximately regenerated.

Example waveforms of the learned softclipping effect are depicted in Fig. 4, where the conditions are as in Sec. 2.4.1. While the dynamics of the learned effects match the **REF** signal roughly, the shape of the

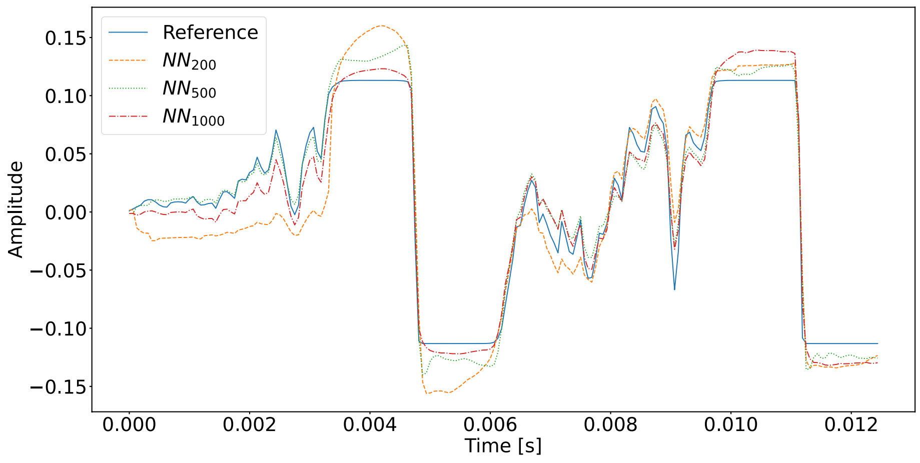


Fig. 4 Waveforms of a section of a **REF** sample processed by the reference softclipping effect with 13 dB gain and the corresponding output signals of **NN200**, **NN500**, and **NN1000**

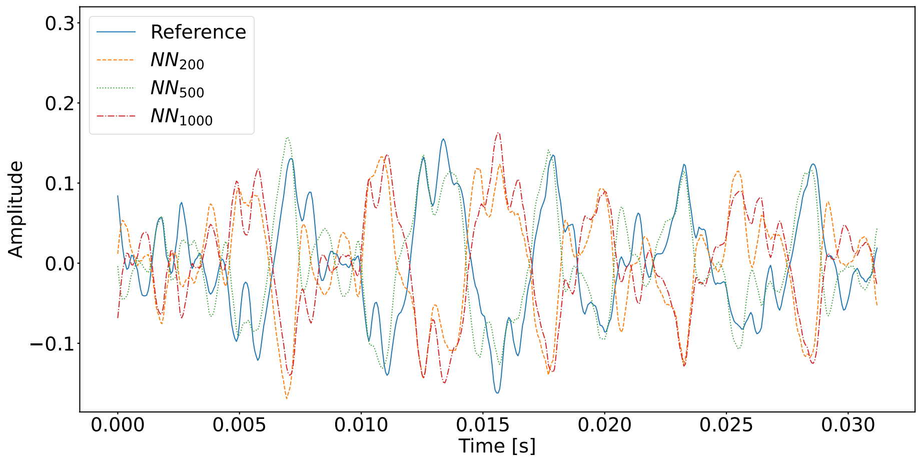


Fig. 5 Waveforms of a section of a **REF** sample processed by the reference phaser effect with setting (1.0, 0.5, 0.5) and corresponding output signals of **NN200**, **NN500**, and **NN1000**

waveforms, especially at the largest peaks, is quite different, resulting in an audible difference in the distribution of the harmonic overtones. This was mentioned to the authors by one of the test subjects. Example waveforms for the learned phaser and slapback delay effect are depicted in Figs. 5 and 6, respectively. For the phaser effect, a phase shift of 180° (polarity inversion) of the waveforms of the **NN200** and **NN1000** condition is apparent, which was occasionally observed throughout our work for different effects. Figure 7 depicts the corresponding spectrograms, which reveal that the phaser effect indeed was learned for the **NN1000** condition and just barely for the **NN500** condition. The **NN200** condition in this case does not exhibit the typical phaser pattern.

3.1 Listening test

Boxplots of the MUSHRA scores are depicted for the softclipping effect in Fig. 8, for the phaser effect in Fig. 9, and for the slapback delay effect in Fig. 10.

The median ratings of the **NN200** and **NN500** condition for the phaser effect are considerably worse than that of the **NN1000** condition due to the fact that the effect is barely or not at all audible for the former two, as can be exemplary seen in Fig. 7. 95% confidence intervals of the median MUSHRA scores, for all test conditions, computed by bootstrapping, are given in Table 4. The

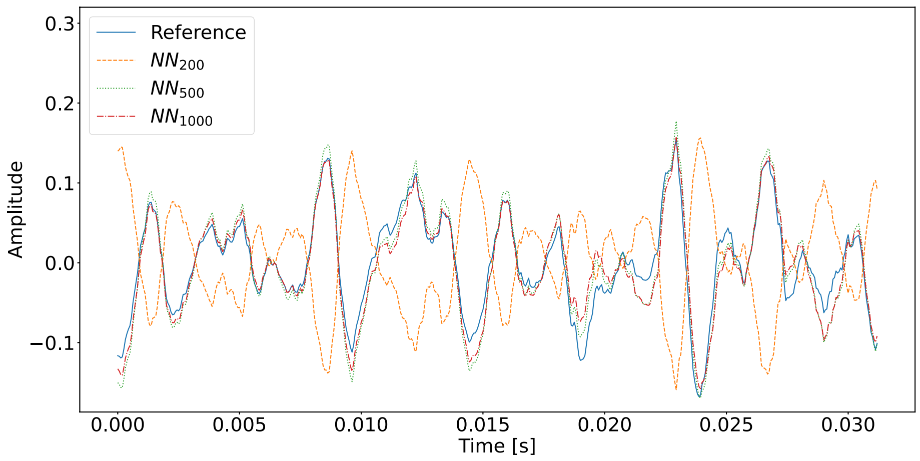


Fig. 6 Waveforms of a section of a **REF** sample processed by the reference slapback delay effect with setting (0.15, 0.5) and corresponding output signals of **NN200**, **NN500**, and **NN1000**

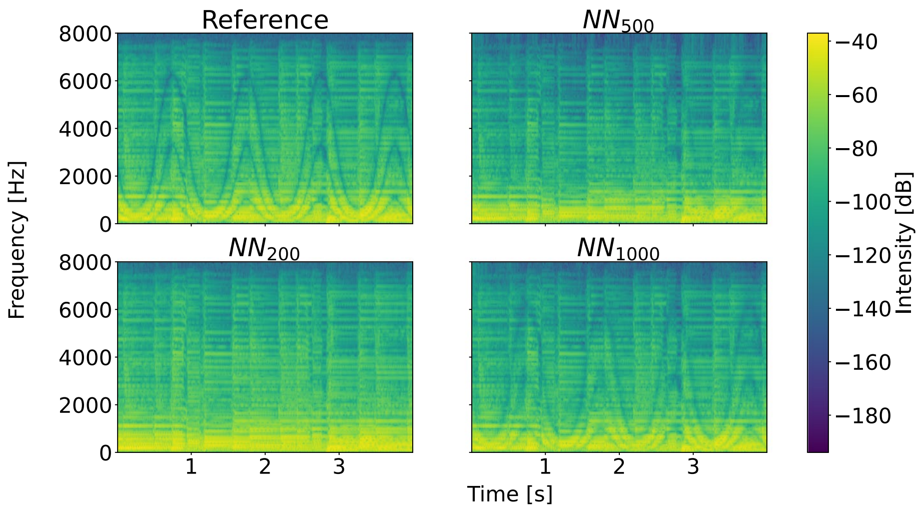


Fig. 7 Spectrograms of a **REF** sample processed by the reference phaser effect with parameter setting (1.0, 0.5, 0.5) and corresponding output signals of **NN200**, **NN500**, and **NN1000**

REF and **ANCHOR** conditions were reliably recognized by the subjects. Outliers are most likely due to fatigue or hard-to-hear settings. For example, the five largest MUSHRA scores of the **ANCHOR** condition in Fig. 8 all occurred for the lowest gain setting of the softclipping effect. A one-way analysis of variance (ANOVA) revealed significant differences between the test conditions for all guitar effects with $F(4, 115) = 334.9\,(p < 0.05)$ for the softclipping effect, $F(4, 115) = 126.3\,(p < 0.05)$ for the phaser effect, and $F(4, 115) = 207.1\,(p < 0.05)$ for the slapback delay effect.

Ten Bonferroni-adjusted Wilcoxon signed-rank tests were performed per effect to investigate pairs of test conditions. The new threshold of significance after Bonferroni-adjustment was $p = 0.005$. Four of these Wilcoxon signed-rank tests compared all other test conditions to the **ANCHOR** condition and revealed significant differences of the medians ($p < 0.001$). The other comparisons and corresponding p-values are listed in Table 5. Except for the phaser effect and the **NN1000** condition, where the **NN1000** condition achieved a perfect median MUSHRA score of 100, for all effects, significant differences were found between the **REF** and the **NN200** to **NN1000** condition. However, this only means that the subjects, on average, identified a difference, not that the quality was poor. For the slapback delay effect, no significant differences between the **NN200** to **NN1000** conditions were observed. There, the **NN200** to

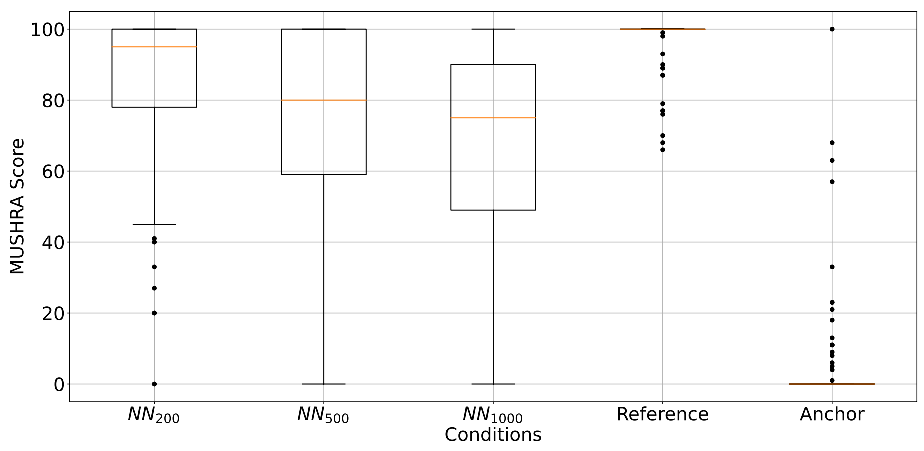


Fig. 8 MUSHRA scores of all three investigated test conditions as well as the **ANCHOR** and **REF** condition for the softclipping effect across all tested effect settings. Anchor and reference were correctly recognized by the subjects with few exceptions, where the five largest outliers for the anchor condition occurred for the weakest softclipping setting, i.e., when the effect was just barely audible

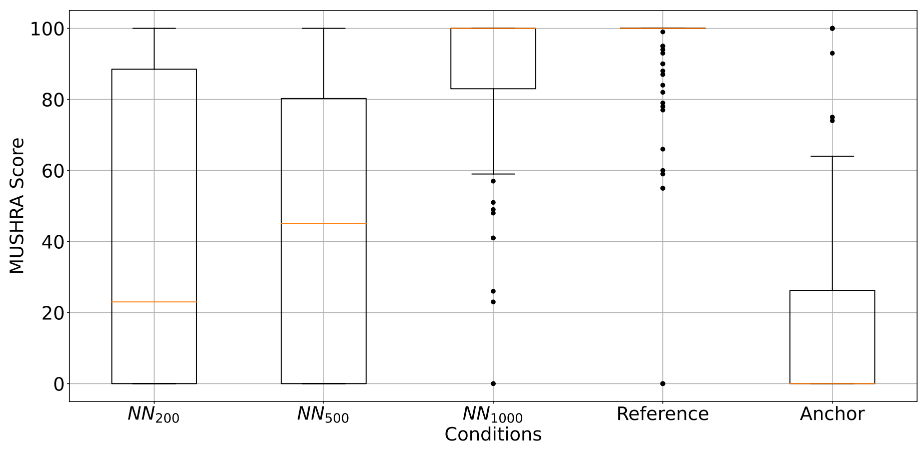


Fig. 9 MUSHRA scores of all three investigated test conditions as well as the **ANCHOR** and **REF** condition for the phaser effect across all tested effect settings. Anchor and reference were correctly recognized by the subjects with few exceptions

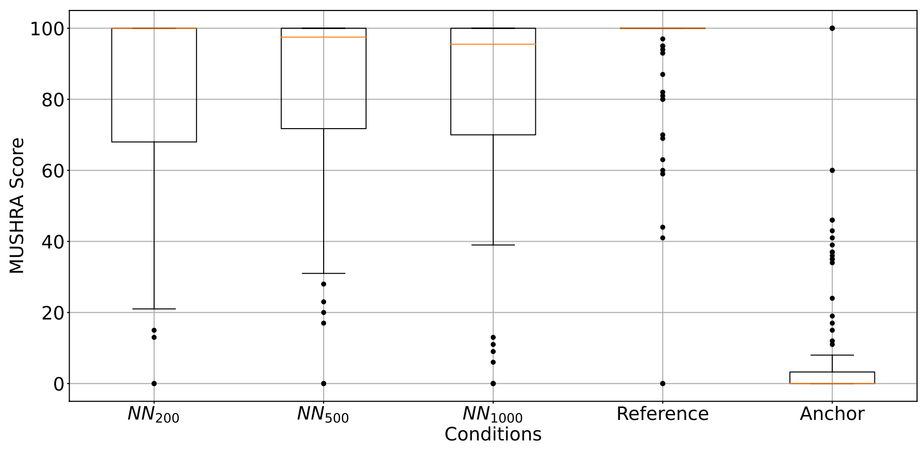


Fig. 10 MUSHRA scores of all three investigated test conditions as well as the **ANCHOR** and **REF** condition for the slapback delay effect across all tested effect settings. Anchor and reference were correctly recognized by the subjects with few exceptions

Table 4 95% confidence intervals of the median MUSHRA score for all test conditions of the listening test across all parameter settings and all guitar effects investigated. Confidence intervals were computed through bootstrapping with 10000 resamples

	NN200	NN500	NN1000	REF	ANCHOR
Softclipping	**93.5** ± 6.5	77.5 ± 7.5	74.0 ± 6.0	100.0 ± 0.0	0.0 ± 0.0
Phaser	21.5 ± 21.5	45.5 ± 14.5	**100.0** ± 0.0	100.0 ± 0.0	0.0 ± 0.0
Slapback delay	**94.25** ± 5.75	94.0 ± 6.0	92.5 ± 7.5	100.0 ± 0.0	0.0 ± 0.0

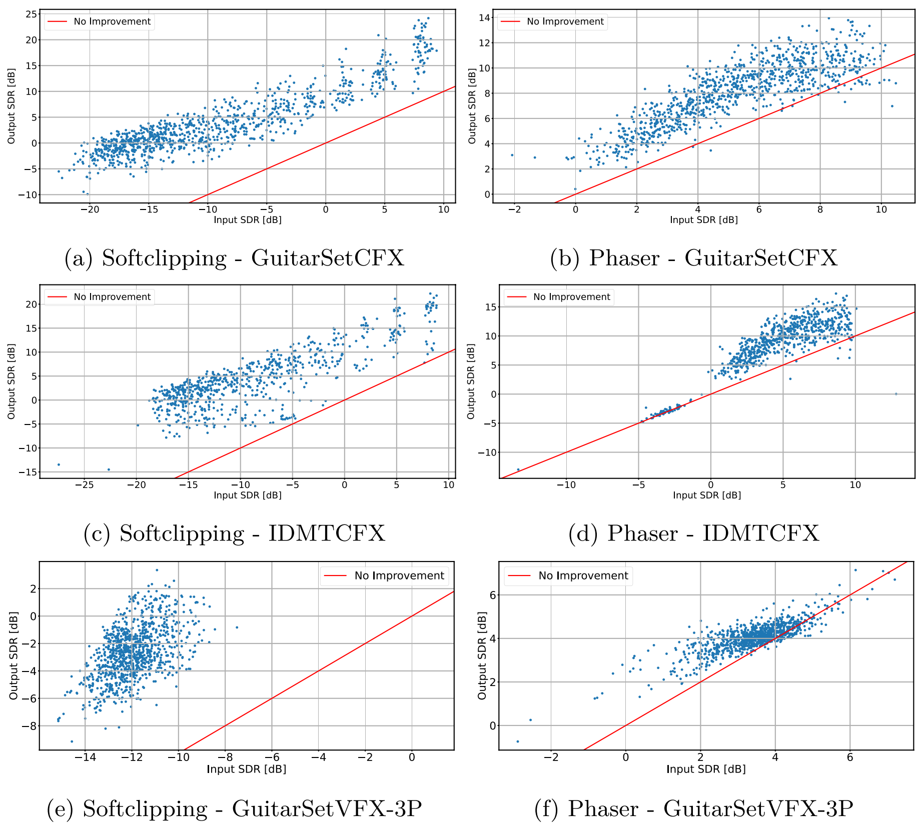
Table 5 Some *p*-values of the Bonferroni-adjusted Wilcoxon-signed-rank-tests for the results of the MUSHRA listening tests. Boldface values indicate statistical significance after applying Bonferroni's correction

Conditions		Effects		
A	B	Softclipping	Phaser	Slapback delay
NN_{200}	NN_{500}	< **0.001**	0.028	0.44
NN_{200}	NN_{1000}	< **0.001**	< **0.001**	0.88
NN_{500}	NN_{1000}	**0.003**	< **0.001**	0.53
NN_{200}	Reference	< **0.001**	< **0.001**	< **0.001**
NN_{500}	Reference	< **0.001**	< **0.001**	< **0.001**
NN_{1000}	Reference	< **0.001**	0.014	< **0.001**

NN1000 conditions achieved median MUSHRA scores of 92.5 and more. In contrast, significant differences were observed for the **NN200** to **NN1000** conditions for the softclipping effect, where the **NN200** achieved superior performance to the **NN500** and **NN1000** condition with a median MUSHRA score of 93.5, compared to 77.5 and 74.0 for the **NN500** and **NN1000** condition, respectively.

3.2 Objective evaluation

For softclipping and phaser, Output SDR across Input SDR for all three test cases as described in Sect. 2.5 is depicted in Fig. 11. For slapback delay, corresponding



(a) Softclipping - GuitarSetCFX (b) Phaser - GuitarSetCFX
(c) Softclipping - IDMTCFX (d) Phaser - IDMTCFX
(e) Softclipping - GuitarSetVFX-3P (f) Phaser - GuitarSetVFX-3P

Fig. 11 Output SDR across Input SDR for HT Demucs processed, i.e., regenerated, samples for softclipping and phaser and the three test cases as described in Sec. 2.5. Note the different *y*-axis scalings

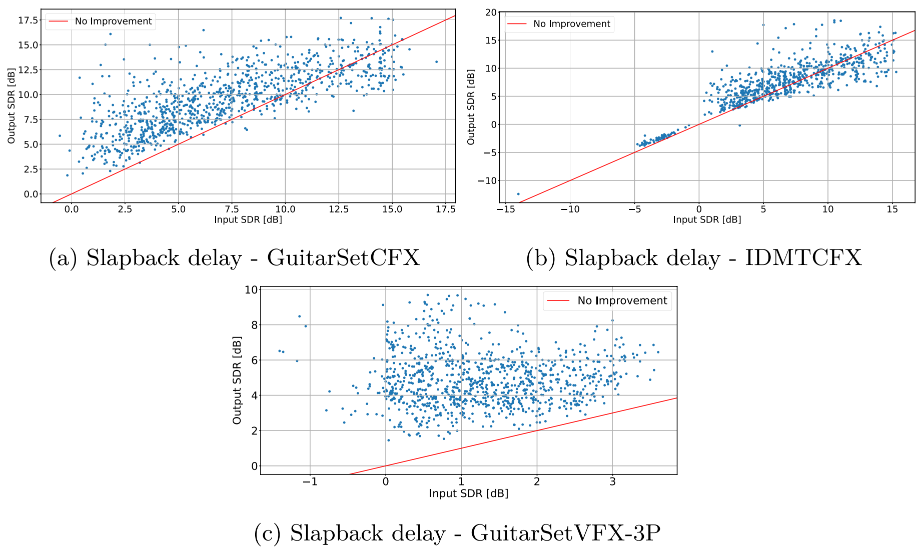


Fig. 12 Output SDR across Input SDR for HT Demucs processed, i.e., regenerated, samples for slapback delay and the three test cases as described in Sec. 2.5. Note the different *y*-axis scalings

Table 6 Overview of the average Output SDR (OSDR), Input SDR (ISDR), and the difference $\Delta SDR := OSDR - ISDR$ for the test cases of the effect removal as described in Sec. 2.5. For ΔSDR, values in parentheses denote standard deviations also in dB

Test case	Effect	Input SDR [dB]	Output SDR [dB]	ΔSDR [dB]
GuitarSetVFX	Softclipping	− 9.7	3.5	13.2 (3.65)
	Phaser	5.1	7.9	2.8 (1.77)
	Slapback delay	7.2	9.6	2.4 (3.2)
IDMTCFX	Softclipping	− 8.8	4.4	13.2 (4.63)
	Phaser	4.6	8.7	4.1 (2.99)
	Slapback delay	6.5	7.3	0.8 (3.62)
GuitarSetVFX-3P	Softclipping	− 10.4	− 2.0	8.4 (2.1)
	Phaser	3.9	4.6	0.7 (1.03)
	Slapback delay	5.4	8.8	3.4 (2.16)

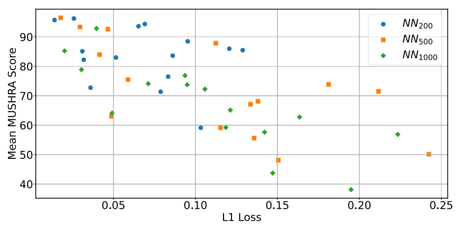
results are shown in Fig. 12. Note the different *y*-axis scalings.

Corresponding averages of Output-SDR (OSDR), Input-SDR (ISDR), and $\Delta SDR := OSDR - ISDR$ for the respective test cases are summarized in Table 6.

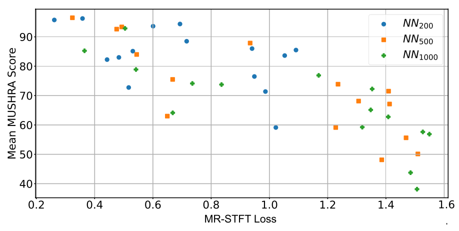
In all scenarios, HT Demucs achieves at least some level of sample regeneration for all three guitar effects. The greatest improvement with respect to SDR was observed for softclipping with an at least 8.4 dB difference in average ΔSDR. Improvements were considerably smaller for phaser and slapback delay, at worst 0.7 dB and 0.8 dB and at best 4.1 dB and 2.4 dB, respectively. Generally, for phaser and slapback delay, higher Input SDR values were observed, almost certainly due to their type of processing, which is not, unlike softclipping, by definition based on signal distortion. Going by SDR, softclipping and phaser signal regeneration appears to generalize well to new guitar signals going by the observed change in Table 6 between the test cases. However, HT Demucs appears to generalize worse to unseen phaser implementations compared to unseen slapback delay implementations as the ΔSDR drops to 0.7 dB for the phaser compared to an increase to 2.4 dB for the slapback delay on the GuitarSetVFX-3P dataset. Going by the results, HT Demucs appears to generalize at least partially to unknown/unseen guitar signals and effect implementations. While the improvement in SDR might suggest a good performance of HT Demucs for softclipping, subjectively, effect removal was actually worst for it on average. The likely cause are the types of signal distortions introduced, which should distort high frequency signal components more than low frequency components. This would explain why high SDRs or high SDR improvements do not necessarily yield corresponding improvements in

Table 7 Correlation coefficients between the MUSHRA scores and the L1 and MR-STFT errors for the conditions of the MUSHRA listening test. In the last row ("Total"), the samples of the different conditions are considered together. $\mathcal{L}$ is the sum of the L1 and MR-STFT errors and is the loss function used for training HT Demucs and the GCNTF networks

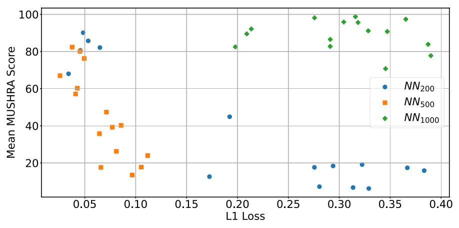
	Softclipping			**Phaser**			**Slapback delay**		
	L1	**MR-STFT**	$\mathcal{L}$	**L1**	**MR-STFT**	$\mathcal{L}$	**L1**	**MR-STFT**	$\mathcal{L}$
NN_{200}	– 0.308	– 0.508	– 0.49	– 0.908	– 0.551	– 0.834	– 0.327	– 0.453	– 0.435
NN_{500}	– 0.668	– 0.804	– 0.794	– 0.833	– 0.813	– 0.825	0.312	– 0.51	– 0.165
NN_{1000}	– 0.8	– 0.788	– 0.797	– 0.155	0.379	0.25	– 0.462	– 0.583	– 0.579
Total	– 0.693	– 0.789	– 0.784	0.035	– 0.738	– 0.658	– 0.021	– 0.518	– 0.38



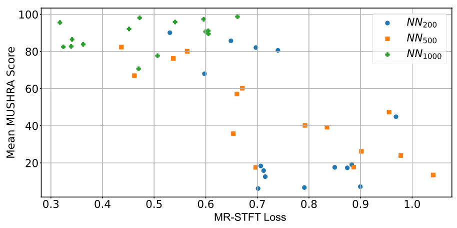
(d) Softclipping



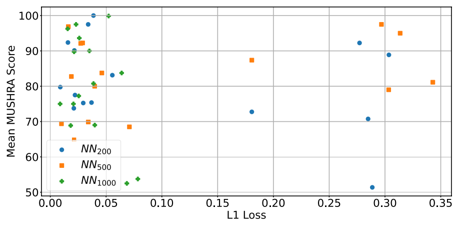
(e) Softclipping



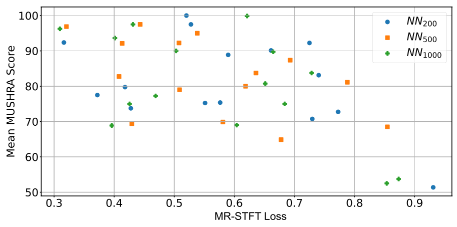
(f) Phaser



(g) Phaser



(h) Slapback delay

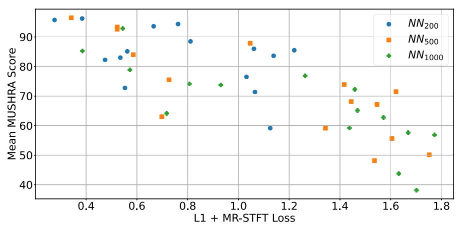


(i) Slapback delay

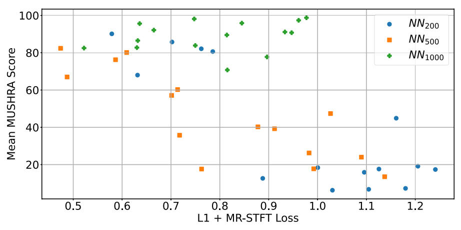
Fig. 13 Distribution of the MUSHRA Scores of the individual samples as a function of the objective metrics. **a**, **b** Softclipping. **c**, **d** Phaser. **e**, **f** Slapback delay

subjective quality. This is supported by a comparison of the MR-STFT loss values on the GuitarCFX dataset. The median MR-STFT loss value for the softclipping is about 2–4 times higher than for phaser and slapback delay.

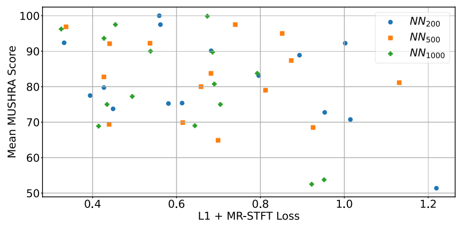
While generally there was a considerable correlation between MUSHRA Scores and corresponding losses as can be seen in Table 7, some interesting exceptions exist and can be observed in Fig. 13: for the phaser and the **NN1000** condition, no clear association of the L1 or MR-STFT loss with the MUSHRA scores exist. Actually, for all phaser samples, the L1 loss was within the range of about 0.2–0.4, similarly for the MR-STFT loss, yet, the corresponding MUSHRA scores appear randomly distributed within a range of about 70–100. An association is clear for the **NN500** condition, which shows a great increase in MUSHRA scores with decreasing L1 loss and similarly an increase in MUSHRA scores with a decrease in MR-STFT loss. The **NN200** condition, while not as erratic with respect to the MUSHRA score-loss association as the **NN1000** condition, shows a considerably



(a) Softclipping



(b) Phaser



(c) Slapback Delay

Fig. 14 Distribution of the MUSHRA Scores of the individual samples as a function of the sum of the objective metrics. **a** Softclipping. **b** Phaser. **c** Slapback delay

less clear connection between MUSHRA scores and loss values. The most erratic behavior was observed for the slapback delay and the L1 loss: most samples for all conditions exhibit an L1 loss in the range of about 0–0.075, yet, the corresponding MUSHRA scores are scattered across a range of about 50 to 100. While still considerably noisy, the corresponding MR-STFT loss values show some linear dependency of the MUSHRA Scores on the MR-STFT losses.

When combining the losses, as done during training, and assessing the relationship between this combined loss to the MUSHRA score, a somewhat less erratic relationship arises as can be seen in Fig. 14. However, even combined, for the phaser and the **NN1000** condition, there does not appear to be a clear pattern in the MUSHRA score-loss scatterplot. A possible explanation could be that the observed differences in MUSHRA scores are due to chance, and for the entire loss range from about 0.5 to about 1.0, the GCNTF achieved about the same quality, which was close to perfect.

4 Discussion

This work investigates blind extraction of guitar effects through blind system inversion followed by neural effect learning. For blind system inversion, we used Hybrid Transformer Demucs (HT Demucs), and for neural effect modeling, we used gated convolutional network with temporal feature-wise linear modulation (GCNTF).

In general, it proved to be possible to blindly copy a given guitar effect, for all effects one of the conditions achieved a median MUSHRA score above 80, which corresponds to excellent quality according to MUSHRA regulations. Unfortunately, there was no consistent best condition, i.e., the number of training samples yielding the best performing model depended on the guitar effect. Additionally, even 200 samples are way too many for a real application, seeing that a music piece of ordinary lengths will cover at most about 3 min of guitar samples, which correspond to about 45 samples of 4 s length. Our current results do not allow to achieve good quality with this few samples, and a lot of research lies ahead to achieve this goal. Even when repeatedly training the GCNTF on a few training samples and testing on these very samples, the GCNTF did not appear to learn the respective guitar effect. For the slapback delay effect, the delayed part of the stimuli signals created by the GCNTF sound slightly smeared in some samples, which is a possible explanation for the outliers depicted in the boxplots. For softclipping, the effect of the GCNTF processed samples sounded too weak if applied only once, as if only half the target gain was applied, and the quality was rated by the authors considerably better after a second processing. While in reality a single pass should suffice, to get an independent judgment for an additional effect, we decided to investigate this two-pass version nonetheless, to see whether softclipping can, in principle, be copied in the proposed way. The cause of this issue certainly is the insufficient clean signal regeneration by Demucs. Since the effect was only partially removed, the GCNTF is merely able to learn the part of the effect that was removed, resulting in a far lower intensity of the learned softclipping effect. For the softclipping effect, the **NN200** condition achieved a significantly higher MUSHRA score than the **NN500** and **NN1000** condition. A hypothesis is that due to passing the input signal twice through the GCNTFs for softclipping as previously mentioned, some additional audable artifacts could arise in the **NN500** and **NN1000** conditions, which are less present in the **NN200**, despite the **NN200** condition yielding inferior performance after a single pass.

For the slapback delay effect, where the **NN200** condition also achieved the highest MUSHRA score, the differences to **NN500** and **NN1000** condition was not significant, and thus it is presumed that the quality of the effect is identical. Seeing that the slapdelay delay effect is simpler than, e.g., the phaser effect, the number of training samples having no apparent impact on the quality of the modeled effect is reasonable. Generally, the effect modeling was found to be rather repeatable, with either yielding identically good effect quality each time or showing only minor variations from training to training. The limiting factor of the proposed approach currently is HT Demucs, which can regenerate a sufficient approximation of the clean input signals only for a handful of guitar effects, and in some of these cases, considerable artifacts still remain. Though, despite these artifacts, sometimes the guitar effects are removed just enough to allow learning them, i.e., perfect removal was found to not be necessary in general. To make it very clear: in all our experiments, given a sufficient effect removal in the input signals, the GCNTFs never failed to yield high quality effect copies. The issue is HT Demucs, not the GCNTF.

4.1 Comparison to other work

Our approach to blindly copy a target guitar effect is closely related to the generative approach by Wright et al. [19], the only other work to blindly model a target guitar effect sound to the best knowledge of the authors. They even achieved very similar performance in their listening test. However, they copy both, effects and timbre of the guitar, while in our approach, the goal is to not copy the timbre of a target guitar but only the applied guitar effect. Depending on the application, this can be an advantage or a disadvantage. The result of our approach is a neural effect model, which could be applied to any guitar without affecting its characteristic timbre. Going by the audio examples provided by Wright et al., their generative approach is generally capable of copying timbre and effect but appears to sometimes modify the sequence of tones, such that the target, reference tone sequence is mixed up and somewhat modified (e.g., higher notes) in the copy. This is a considerable disadvantage for the aim of allowing to copy a sound of a guitar. In that aspect, our approach is considerably superior as this cannot happen due to the method we apply. Their approach and ours appears to require a similar amount of data to yield sufficient performance; however, due to our use of HT Demucs, our approach would be considerably more computationally complex.

4.2 Limitations

In this work, we only investigated blind extraction of guitar effects using guitar only samples, i.e., no other instruments were present in the recordings. Furthermore, only one implementation per guitar effect was used. While we did a few successful tests with instrument mixes and different implementations, the performance is far from sufficient and a lot of further work is required to yield useful results. Some general limits or constraints exist to our approach: the guitar input signal of a target guitar effect has to be sufficiently rich in spectral content to allow blind extraction of an "entire" effect. This is due to the regeneration of the clean input signal and is known from blind system identification theory [35]. Examples can be easily thought up, e.g., a guitar effect that acts differently depending on the frequency content of the input signal processing a signal which only contains a certain frequency region. Some, currently unknown, key characteristics of the clean input guitar signal have to be left unmodified by a target guitar effect to allow to estimate the input signal. However, how much these limits will actually prohibit blind effect copies is not clear as, e.g., no signal model[1] exists for guitars. Furthermore, the proposed system does not allow the tweaking of effect parameters, meaning that learned effects cannot be modified by hand afterwards. If a modification of the sound of the learned effect is desired, additional pre- or postprocessing (e.g., equalization) has to be introduced afterwards. This limitation is shared with the approach of Wright et al. [19]. Allowing to tweak the sound of the learned model could be perhaps achieved using pretrained effect embeddings and a powerful, pretrained effect modeling network. Modulating extracted embeddings could then allow modulating the copied guitar sound[2].

4.3 Future work

One obvious research endeavor is the improvement of HT Demucs or the development of other approaches, specifically designed for effect removal, seeing that HT Demucs is currently the bottleneck of the proposed approach. Additionally, to potentially allow subsequent modification of the learned effect, disentangled autoencoders could be applied, similar to approaches to voice

[1] In the sense of, e.g., ratios of overtones and rate of decay among others distinguishing guitars from other instruments.

[2] Idea due to our colleague Lars Rumberg.

conversion [36]. If a sufficiently powerful autoencoder was used to process a guitar signal that has been processed by some guitar effect, and the said autoencoder yielded a disentangled latent representation of the guitar signal which separates the original, clean guitar signal from the effect processing used to compute the processed guitar signal, subsequent modification of the effect sound might be feasible. In the latent space, the components corresponding to the clean guitar signal could be replaced by a new guitar signal to be processed, while the components of the latent space corresponding to the guitar effect could be modified in a desired way.

5 Conclusions

This work investigates blind extraction of guitar effects using Hybrid Transformer Demucs for blind system inversion and gated convolutional network with temporal feature-wise linear modulation for neural guitar effect modeling. The proposed method is tested with the phaser, softclipping, and slapback delay effect. Listening tests with eight subjects indicated excellent quality of the the blind copies, i.e., little to no difference to the reference guitar effect.

Acknowledgements
The authors would like to thank all subjects that participated in our study.

Authors' contributions
R.H.: idea, main author, study design. K.G.: idea refinement, implementation, study design and execution, proof-reading. A.L.: proof-reading. J.O.: supervision, proof-reading. R.H. and K.G. declare equal contribution.

Funding
Open Access funding enabled and organized by Projekt DEAL. The research has not been funded by third parties.

Availability of data and materials
All stimuli of the MUSHRA test and additional audio material can be found under https://rhtnt.github.io/BlindEffectExtraction/ or https://seafile.cloud.uni-hannover.de/d/2aafe578ae7b4bb0beaa/.

Declarations

Ethics approval and consent to participate
All subjects gave their informed consent to participate. All subjects agreed to anonymous usage of their results.

Consent for publication
All subjects gave their informed consent to anonymous usage of their data and results in scientific publications.

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

Received: 30 June 2023 Accepted: 22 January 2024
Published online: 07 February 2024

References

1. T. Wilmering, D. Moffat, A. Milo, M. Sandler, A history of audio effects. Appl. Sci. **10**, 791 (2020). https://doi.org/10.3390/app10030791
2. A. Sarti, U. Zoelzer, X. Serra, M. Sandler, S. Godsill, Digital audio effects. EURASIP J. Adv. Signal Proc. **2010**(1), 459654 (2011). https://doi.org/10.1155/2010/459654
3. U. Zölzer, *DAFX: Digital Audio Effects*, 2nd edn. (Wiley, Chichester, 2011)
4. A.P. McPherson, J.D. Reiss, *Audio Effects: Theory, Implementation and Application* (CRC Press, Boca Raton, 2014)
5. M. Stein, J. Abeßer, C. Dittmar, G. Schuller, *Automatic detection of audio effects in guitar and bass recordings* (J. Audio Eng, Soc, 2010)
6. H. Jürgens, R. Hinrichs, J. Ostermann, in *Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx2020)*, Recognizing guitar effects and their parameter settings (2020). https://www.dafx.de/paper-archive/2020/proceedings/papers/DAFx2020_paper_2.pdf. Accessed 30 Jan 2024
7. R. Hinrichs, K. Gerkens, J. Ostermann, in *11th International Conference on Artificial Intelligence in Music, Sound, Art and Design (EvoMUSART)*, Springer, Classification of guitar effects and extraction of their parameter settings from instrument mixes using convolutional neural networks (2022). https://doi.org/10.1007/978-3-031-03789-4_7
8. R. Hinrichs, K. Gerkens, A. Lange, J. Ostermann, Convolutional neural networks for the classification of guitar effects and extraction of the parameter settings of single and multi-guitar effects from instrument mixes. EURASIP J. Audio Speech Music Process. **2022**(1) (2022). https://doi.org/10.1186/s13636-022-00257-4
9. M. Stein, in *13th International Conference on Digital Audio Effects, DAFx 2010 Proceedings*, Automatic detection of multiple, cascaded audio effects in guitar recordings (2010). https://dafx.de/paper-archive/2010/DAFx10/Stein_DAFx10_P18.pdf. Accessed 30 Jan 2024
10. F. Eichas, M. Fink, U. Zölzer, in *Proceedings of the 18th International Conference on Digital Audio Effects (DAFx-15)*, Feature design for the classification of audio effect units by input/ output measurements (2015). https://www.dafx.de/paper-archive/2015/DAFx-15_submission_6.pdf. Accessed 30 Jan 2024
11. M. Schmitt, B. Schuller, in *INFORMATIK 2017*, Recognising guitar effects - which acoustic features really matter? (Gesellschaft für Informatik, Bonn, 2017), pp. 177–190. https://doi.org/10.18420/in2017_12
12. N. Masuda, D. Saito, in *Proceedings of the 24rd International Conference on Digital Audio Effects (DAFx2021)*, Quality diversity for synthesizer sound matching (2021). https://dafx2020.mdw.ac.at/proceedings/papers/DAFx20in21_paper_46.pdf. Accessed 30 Jan 2024
13. M.J. Yee-King, L. Fedden, M. d'Inverno, Automatic programming of VST sound synthesizers using deep networks and other techniques. IEEE Trans. Emerg. Top. Comput. Intell. **2**(2), 150–159 (2018). https://doi.org/10.1109/TETCI.2017.2783885
14. D. Sheng, G. Fazekas, in *Proceedings of the 20rd International Conference on Digital Audio Effects (DAFx2017)*, Automatic control of the dynamic range compressor using a regression model and a reference sound (2017). https://dafx17.eca.ed.ac.uk/papers/DAFx17_paper_44.pdf. Accessed 30 Jan 2024
15. D. Sheng, G. Fazekas, in *2019 International Joint Conference on Neural Networks (IJCNN)*, A feature learning siamese model for intelligent control of the dynamic range compressor (2019), pp. 1–8. https://doi.org/10.1109/IJCNN.2019.8851950
16. M. Comunitá, D. Stowell, J.D. Reiss, Guitar effects recognition and parameter estimation with convolutional neural networks. J. Audio Eng. Soc. **69**(7/8), 594–604 (2021)
17. S. Lee, J. Park, S. Paik, K. Lee, in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Blind estimation of audio processing graph (2023), pp. 1–5. https://doi.org/10.1109/ICASSP49357.2023.10096581
18. C. Peladeau, G. Peeters. Blind estimation of audio effects using an auto-encoder approach and differentiable signal processing (2023), arXiv:2310.11781
19. A. Wright, V. Välimäki, L. Juvela, in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Adversarial guitar amplifier modelling with unpaired data (2023), pp. 1–5. https://doi.org/10.1109/ICASSP49357.2023.10094600

20. J. Imort, G. Fabbro, M.A.M. Ram'irez, S. Uhlich, Y. Koyama, Y. Mitsufuji, *in International Society for Music Information Retrieval Conference* (Learning how to recover the clean signal, Distortion audio effects, 2022)
21. M. Rice, C.J. Steinmetz, G. Fazekas, J.D. Reiss, in *2023 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA),* General purpose audio effect removal (2023), pp. 1–5. https://doi.org/10.1109/WASPAA58266.2023.10248157
22. M. Comunitá, C.J. Steinmetz, H. Phan, J.D. Reiss, in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Modelling black-box audio effects with time-varying feature modulation (2023), pp. 1–5. https://doi.org/10.1109/ICASSP49357.2023.10097173
23. A. Défossez, N. Usunier, L. Bottou, F. Bach. Music source separation in the waveform domain (2021), arXiv:1911.13254
24. A. Défossez. Hybrid spectrogram and waveform source separation (2022), arXiv:2111.03600
25. S. Rouard, F. Massa, A. Défossez, in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Hybrid transformers for music source separation (2023), pp. 1–5. https://doi.org/10.1109/ICASSP49357.2023.10096956
26. T. Vanhatalo, P. Legrand, M. Desainte-Catherine, P. Hanna, A. Brusco, G. Pille, Y. Bayle, A review of neural network-based emulation of guitar amplifiers. Appl. Sci. **12**(12) (2022). https://doi.org/10.3390/app12125894
27. A. Wright, E.P. Damskägg, V. Välimäki, in *Proceedings of the 22nd International Conference on Digital Audio Effects (DAFx-19), Birmingham, UK, September 2–6, 2019*, Real-time black-box modelling with recurrent neural networks (2019). https://dafx.de/paper-archive/2019/DAFx2019_paper_43.pdf. Accessed 30 Jan 2024
28. Facebook. Demucs music source separation. Github, https://github.com/facebookresearch/demucs. Accessed 18 Mar 2023
29. Q. Xi, R.M. Bittner, J. Pauwels, X. Ye, J.P. Bello, in *International Society for Music Information Retrieval Conference*, Guitarset: a dataset for guitar transcription (2018). https://archives.ismir.net/ismir2018/paper/000188.pdf. Accessed 30 Jan 2024
30. Spotify AB. Pedalboard (2022). GitHub, https://github.com/spotify/pedalboard. Accessed 20 June 2023
31. R. Yamamoto, E. Song, J.M. Kim. Probability density distillation with generative adversarial networks for high-quality parallel waveform generation (2019), arXiv:1904.04472
32. C.J. Steinmetz, J.D. Reiss, in *Digital Music Research Network One-day Workshop (DMRN+15)*, Auraloss: Audio focused loss functions in PyTorch (2020). https://static1.squarespace.com/static/5554d97de4b0ee3b50a3ad52/t/5fb1e9031c7089551a30c2e4/1605495044128/DMRN15__auraloss__Audio_focused_loss_functions_in_PyTorch.pdf. Accessed 30 Jan 2024
33. M. Schoeffler, S. Bartoschek, F.R. Stöter, M. Roess, S. Westphal, B. Edler, J. Herre, webMUSHRA — a comprehensive framework for web-based listening tests. J. Open Res. Softw. (2018). https://doi.org/10.5334/jors.187
34. ITU-R, Algorithms to measure audio programme loudness and true-peak audio level. Tech. Rep. BS.1770-4 (International Telecommunication Union, 2015), https://www.itu.int/dms_pubrec/itu-r/rec/bs/R-REC-BS.1770-4-201510-I!!PDF-E.pdf
35. K. Abed-Meraim, W. Qiu, Y. Hua, Blind system identification. Proc. IEEE **85**(8), 1310–1322 (1997). https://doi.org/10.1109/5.622507
36. C. Chan, K. Qian, Y. Zhang, M. Hasegawa-Johnson, in *2022 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2022 - Proceedings*, Speechsplit2.0: unsupervised speech disentanglement for voice conversion without tuning autoencoder bottlenecks (United States, 2022), pp. 5243–5247. https://doi.org/10.1109/ICASSP43922.2022.9747763