OPEN ACCESS Freely available online
M. Comunità, D. Stowell, and J. D. Reiss, "Guitar Effects Recognition and Parameter Estimation With Convolutional Neural Networks"
J. Audio Eng. Soc., vol. 69, no. 7/8, pp. 594–604, (2021 July/August).
DOI: https://doi.org/10.17743/jaes.2021.0019

Guitar Effects Recognition and Parameter Estimation With Convolutional Neural Networks

MARCO COMUNITÀ,[*] **DAN STOWELL, AND JOSHUA D. REISS,** *AES Fellow*
(m.comunita@qmul.ac.uk) (dan.stowell@qmul.ac.uk) (joshua.reiss@qmul.ac.uk)

Centre for Digital Music, Queen Mary University of London, UK

Despite the popularity of guitar effects there is very little existing research on classification and parameter estimation of specific plugins or effect units from guitar recordings. In this paper, convolutional neural networks were used for classification and parameter estimation for 13 overdrive, distortion, and fuzz guitar effects. A novel dataset of processed electric guitar samples was assembled, with four sub-datasets consisting of monophonic or polyphonic samples and discrete or continuous settings values, for a total of about 250 hours of processed samples. Results were compared for networks trained and tested on the same or a different sub-dataset. We found that discrete datasets could lead to equally high performance as continuous ones while being easier to design, analyze, and modify. Classification accuracy was above 80%, with confusion matrices reflecting similarities in the effects timbre and circuits design. With parameter values between 0.0 and 1.0, the mean absolute error is in most cases below 0.05, while the root mean square error is below 0.1 in all cases but one.

0 INTRODUCTION

In music composition, production, and engineering, audio effects play an essential role in altering the sound toward the desired final result. For instruments like the electric guitar, the processing signal chain can often be viewed as part of the artist's creative expression [1]. Entire musical genres and styles are frequently defined and identified by the type of audio effects adopted [2, 3] and renowned musicians commonly rely on specific combinations of guitars, amplifiers, and effects to achieve a unique sound [4]. Through the decades, artists, engineers, and producers have defined a palette of sounds that have become a reference for guitar players. In the effort to recreate a specific sound or atmosphere, professionals and amateurs go to great lengths to identify the exact gear that was used in a certain recording. When describing a desired result people often rely on naming a reference style, artist, or song rather than talking in terms of sound features or effect parameters.

Although the design, reconstruction, and emulation of audio effects is well-studied [5, 6], it is less so for their recognition and parameter estimation. Therefore, in our work, we set out to develop a deep learning model capable of recognizing which specific guitar pedal effect was used in a recording as well as estimating its parameters. Being the single most important effect for electric guitar, and the one that usually triggers most discussions, this work focused on nonlinear effects, i.e., overdrive, distortion, and fuzz.

The rest of this paper is organized as follows: in Sec. 1 we look at the state of the art and work related to guitar effects recognition, Sec. 2 introduces the dataset we assembled for this work while the networks architecture is described in Sec. 3, Secs. 4 and 5 outline experiments and results, and Secs. 6 and 7 are devoted to discussion and conclusions.

1 BACKGROUND

The recognition of musical instrument sounds has been of interest to the information retrieval community for a long time [7], with applications in musical sounds databases, intelligent music search, and recommendation systems. The estimation of instruments' parameters, as well as the classification of playing gestures or styles, has also been extensively studied—and applied in contexts like automatic music transcription [8, 9], music education [10], or musicology [11].

Many papers focused on guitar parameters. String, fret, and plucking position estimation have been extensively studied, including works on classical and acoustic [12] and electric [13] guitar. Most works are based on features

[*]To whom correspondence should be addressed

extracted from recorded sounds [9, 12–15] but there are also examples of estimation based on guitar-string physical models [16, 17] for high-tempo and real-time applications. Plucking and pickup position estimation has also been studied, with Mohamad et al. exploring solutions based on spectral features (comparing recordings with string models) and autocorrelation [18]. The study was also extended to the case of nonlinear audio effects in the signal chain [19].

Classification of playing styles and techniques has also received substantial attention. In [20] the authors compare the performance of several classifiers (support vector machine (SVM), Gaussian mixture models, nearest neighbors) on 5 plucking styles (e.g., finger, pick, slap) from bass guitar recordings. In [21], Schuller et al. extend the work to include expression styles (e.g., bend, slide, vibrato). Su et al., in [22], apply sparse dictionary learning to classify guitar playing techniques (e.g., vibrato, hammer-on, pull-off). The same classification problem is solved in [23] using a deep belief network.

Other examples of work on guitar-related classification problems focused on playing mode (e.g., bass, solo melodic improvisation, chordal playing) [24], chords fingering [25], and guitar model [26–29]. However there is only a small corpus of research on guitar effects recognition [30–34].

In [30], Stein worked with guitar and bass recordings on recognition of 11 different effects: feedback and slapback delay, reverb, chorus, flanger, phaser, tremolo, vibrato, distortion, and overdrive. In a subsequent work [31] the author extended his method—based on spectral, cepstral, and harmonic features and SVMs—to cascaded effects. In [33], using the same dataset, the authors aimed to understand which are the most relevant features for the classification task adopting a “bag-of-audio-words” approach, while in [32], the authors—making use of specific input test signals and extracting features from the output—worked on classification of 10 analogue effect units into 5 categories. In [35] the guitar amplifier modeling process includes emulation of nonlinear blocks and estimation of parameters. In these last examples the approach is limited to the case in which the unit to classify/model is accessible, which defeats the purpose of estimation from recordings.

The closest study to our work is [34], and it is the only one that estimates effects parameters from guitar recordings. Similarly to the previous studies, the authors used SVMs to classify the same 11 effects listed above with a reduced features set and extended the work by training shallow neural networks on parameter estimations for 3 effects (distortion, tremolo, delay). The main limitation of this study is the necessity of separate features selection and network training for parameter estimation on each effect.

In all these cases the authors worked on generic audio effects (e.g., compressor, tremolo, delay) or categories (e.g., filter, ambience, modulation, nonlinear) and, to the best of our knowledge, there is no previous research focusing on classification and parameter estimation of specific plugins or effect units (see Table 1) from guitar recordings. In our work, conditioning the estimation network on the effect class allows one to infer the settings’ values without the need for different input representation, network architectures, or separate training.

Table 1. Plugins.

Designer	Plugin	Emulation of	Id
Audified	MultiDrive Pedal Pro	Ibanez TS808	808
		Ibanez TS9	TS9
		Boss BD2	BD2
		Boss OD1	OD1
		Boss SD1	SD1
		Boss DS1	DS1
		ProCo Rat	RAT
		MXR Distortion+	DPL
		Arbiter Fuzz Face	FFC
		EH Big Muff	BMF
Mercuriall	Greed Smasher	Mesa/Boogie Grid Slammer	MGS
Analog Obsession	Pig Pie	EH Russian Big Muff	RBM
	Zupaa	Vox Tone Bender	VTB

2 DATASET

We assembled a novel dataset of processed electric guitar samples using unprocessed recordings from the IDMT-SMT-Audio-Effects dataset [30]. The source dataset[1] includes monophonic (624 single notes) and polyphonic (420 intervals and chords) recordings (wav, 44.1 kHz, 16 bit, mono) from two different electric guitars, each with two pick-up settings and up to three plucking styles. The monophonic recordings cover the common pitch range of a six-string electric guitar, and the polyphonic samples were obtained mixing single notes recordings to generate two-note intervals and three- or four-note chords. All samples are 2 s long. The monophonic recordings required removal of background noise before the note onset, which we obtained using a python script together with *Librosa*’s [36] onset detection function.

To assemble our dataset we selected 13 overdrive, distortion, and fuzz plug-ins (see Table 1) designed to emulate some of the most iconic and widely used analogue guitar effect pedals. All the plugins have two or three controls and, regardless of the specific name adopted by the designer, the controls can be identified by their processing function: Level, Gain, Tone/Equalization.

For training and testing purposes four sub-datasets were generated, which will be referred to as Mono Discrete, Poly Discrete, Mono Continuous, and Poly Continuous. The first two subsets (Mono Discrete, Poly Discrete) use a discrete set of combinations selected as the most common and representative settings a person might use: Gain = [0.0, 0.1, 0.2, 0.5, 0.8, 1.0], Tone/Eq = [0.0, 0.2, 0.5, 0.8, 1.0]. Also, since the Level control has no effect on the output timbre, it was set to 1.0 for every combination. A summary of the controls and settings is shown in Table 2. Most plugins do not include Gain values below 0.2—this is

[1] https://www.idmt.fraunhofer.de/en/business_units/m2d/smt/audio_effects.html

Table 2. Settings.

Id	Level	Gain	Tone/Eq
808	[1.0]	[0.2, 0.5, 0.8, 1.0]	[0.0, 0.2, 0.5, 0.8, 1.0]
TS9	[1.0]	[0.2, 0.5, 0.8, 1.0]	[0.0, 0.2, 0.5, 0.8, 1.0]
BD2	[1.0]	[0.2, 0.5, 0.8, 1.0]	[0.0, 0.2, 0.5, 0.8, 1.0]
OD1	[1.0]	[0.2, 0.5, 0.8, 1.0]	···
SD1	[1.0]	[0.2, 0.5, 0.8, 1.0]	[0.0, 0.2, 0.5, 0.8, 1.0]
DS1	[1.0]	[0.2, 0.5, 0.8, 1.0]	[0.0, 0.2, 0.5, 0.8, 1.0]
RAT	[1.0]	[0.2, 0.5, 0.8, 1.0]	[0.0, 0.2, 0.5, 0.8, 1.0]
DPL	[1.0]	[0.2, 0.5, 0.8, 1.0]	···
FFC	[1.0]	[0.0, 0.2, 0.5, 0.8, 1.0]	···
BMF	[1.0]	[0.2, 0.5, 0.8, 1.0]	[0.0, 0.2, 0.5, 0.8, 1.0]
MGS	[1.0]	[0.2, 0.5, 0.8, 1.0]	[0.0, 0.2, 0.5, 0.8, 1.0]
RBM	[1.0]	[0.2, 0.5, 0.8, 1.0]	[0.0, 0.2, 0.5, 0.8, 1.0]
VTB	[1.0]	[0.1, 0.2, 0.5, 0.8, 1.0]	···

Table 3. Architecture.

Layer	Size	#Fmaps	Activation
Convolution 2D	5x5	6	Linear
Batch Normalization	···	···	···
Activation	···	···	ReLU
Max Pooling	2x2	···	···
Convolution 2D	5x5	12	Linear
Batch Normalization	···	···	···
Activation	···	···	ReLU
Max Pooling	2x2	···	···
Fully Connected	120	···	Linear
Batch Normalization	···	···	···
Activation	···	···	ReLU
Fully Connected	60	···	Linear
Batch Normalization	···	···	···
Activation	···	···	ReLU
Fully Connected	*	···	**

Trainable Parameters: FxNet/SetNet ≈ 760 k, SetNetCond ≈ 1.3 M, MultiNet ≈ 1.5 M
*FxNet = #Plug-ins - SetNet = #Settings
**FxNet = Linear - SetNet = Tanh

because for such values there is no audible change between input and output or even a level attenuation (with no output for Gain = 0.0). Every monophonic and polyphonic sample was processed with all the combinations, generating a total of ∼200,000 processed samples (∼120,000 monophonic, ∼80,000 polyphonic), for a total of about 110 hours.

For the second two subsets (Mono Continuous, Poly Continuous), both unprocessed samples as well as settings' values were drawn from a uniform distribution. We generated 10,000 random samples for each plugin, obtaining a total of 260,000 samples (130,000 monophonic, 130,000 polyphonic), equivalent to about 140 hours. Settings values were limited to fall between the extremes shown in Table 2.

Generating these four subsets we aimed at gaining a deeper understanding about the generalization capabilities of our models. The samples' were processed in MATLAB—making use of its VST plugin host features—and both unprocessed inputs and processed outputs were normalized to −6 dBFS.

3 ARCHITECTURE

Our neural network architecture (Table 3) is based on a combination of two convolutional and three fully connected layers, with batch normalization layers at each hidden level. Except for the output layers' size and activation functions, the same configuration was used to train networks on both effects classification and settings estimation. Four different networks—and training paradigms—were implemented:

- Effects classification network (FxNet)
- Settings estimation network (SetNet)
- Multitask classification and estimation network (MultiNet)—where the two convolutional layers are shared
- Settings estimation conditional network (SetNetCond)—where an extra embedding layer is added to condition the estimation on the effect class

The loss functions adopted for the classification and estimation problems were, respectively, the cross-entropy loss and mean square error (MSE). To evaluate the settings estimation networks we also defined an accuracy metric for which a prediction is considered correct when the absolute error for every parameter is less than 0.1. For effects that do not have a Tone/Eq control we represented the absence using a value of −1.0 for the prediction. The threshold of 0.1 was chosen to simplify the comparison of networks performance in different training settings and on different datasets. The value is based on the authors' experience and informal listening tests during the dataset creation phase. However, parameters' sensitivity varies between effects and controls, and differences in absolute value often do not relate linearly with perceptual differences. To overcome these limitations we do also rely on mean absolute error (MAE) and root mean square error (RMSE) to evaluate our models.

As input features to all our networks we used mel power-spectrograms, extracted from audio in 23 ms Hann windows with 50% overlap. In total 128 mel-bands were used in the 0–22,050 Hz range. For a given 2 s audio input, the feature extraction produced a T x 128 output (T = 87). These features are motivated by human auditory perception and are a common choice in acoustic scene classification [37]. Due to the good performance of our models, we did not deem it important for this study to test other features, but it could be worth exploring in the future.

4 EXPERIMENTS

Some preliminary experiments were conducted to obtain baseline performances for the settings estimation problem and compare the results when using a multitask approach or conditional network. The literature shows how multitask learning can be effective at solving related tasks [38] and more efficient than training several networks. In the multitask paradigm the network was trained to classify a sample and estimate its settings at the same time. In the conditional paradigm the networks were trained in "series," with the classification network (FxNet) used to condition the set-

Table 4. FxNet accuracy (%).

	Test			
Train	Mono Disc.	Mono Cont.	Poly Disc.	Poly Cont.
Mono Disc.	86.3	83.1	···	···
Mono Cont.	81.3	90.9	···	···
Poly Disc.	···	···	88.4	89.4
Poly Cont.	···	···	84.1	91.4

tings estimation network (SetNetCond). The experiments were conducted on the Mono Discrete dataset.

All networks were trained for 50 epochs, which resulted in the following test accuracy:

- SetNet = 40.3%
- MultiNet = 40.88% (87.0% classification accuracy, 44.6% estimation accuracy)
- FxNet + SetNetCond = 57.3% (89.7% classification accuracy, 60.7% estimation accuracy)

The results show no appreciable difference in effect classification accuracy between the multitask and conditional paradigms but do show an impact on the settings estimation accuracy, with the conditional network performing better. Further experiments were therefore centered on the analysis of the classification network (FxNet) and the conditional estimation network (SetNetCond) when trained/tested on the four different sub-datasets. In the following section we illustrate the results of training the networks for 100 epochs with early stopping when the validation accuracy sees no improvement for 15 epochs. For weights update we used the Adam optimizer [39] with a fixed learning rate of 0.001.

5 RESULTS

5.1 Effects Classification

Table 4 shows the accuracy results for the effect classification problem. Note that the test accuracy is higher for networks trained on continuous datasets, respectively 90.9% for the Mono Continuous dataset and 91.4% for the Poly Continuous dataset. At the same time, networks trained on discrete datasets performed better when tested on continuous ones than the opposite condition (networks trained on continuous and tested on discrete datasets): 83.1% vs 81.3% for monophonic samples and 89.4% vs 84.1% for polyphonic ones.

By analyzing the confusion matrices we can gain a better understanding about the networks' performance as well as the challenges behind the classification. Fig. 1 shows the details for networks trained on discrete datasets. About 10% of the errors in both datasets are due to misclassification between 808 and TS9. The plugins are emulations of two overdrive effects from the same manufacturer (Ibanez TS808 and TS9). The two effects are supposed to have similar gain and frequency response, but upon studying the circuits schematics we did not notice any difference between the two. Hence our network would confuse samples from either effect.

An explanation for the misclassification imbalance between monophonic and polyphonic samples might be related to the training procedure. We used batches of 100 samples, randomly selected across the whole dataset and without control for batch to dataset ratio over the 13 classes. Assuming identical plugins, the network will tend to classify samples from either as belonging to the class that was seen first or more during training. To support this hypothesis we retrained the FxNet network excluding TS9 samples from the dataset, reaching 96.9% and 98.7% accuracy for monophonic and polyphonic samples, respectively, with 100% accuracy on 808 samples.

Similar observations are valid for errors in classifying OD1 and SD1. Again, the plugins are emulations of effect pedals from the same manufacturer and in their original analogue version use very similar designs. The two share the same clipping circuit, although while the SD1 includes a tone control that combines a treble boosting first order shelving filter with a first order lowpass, the OD1 has no tone control and a fixed first order lowpass.

Analyzing the classification errors for the Mono Discrete dataset, we noticed how, of the 345 times the OD1 is classified as SD1: 31% of the times is when the Gain control is set to 0.2 and another 31% when Gain = 0.5. Ignoring the specific note being played, this result might be a consequence of low gain settings, where the spectral differences might be too small. On the other hand the SD1 is confused for the OD1 88 times and all cases are from samples where the Tone is set to 0, 0.2, or 0.5. For low Tone values the spectral differences might be unnoticeable; most of the high frequency harmonics might be filtered. In this case we did not observe correlation with the Gain control values.

Another interesting example more noticeable for the Mono Discrete dataset is the misclassification of the DPL as RAT. In this case we are referring to effects from different manufacturers that do share some circuit design choices. Although the DPL does not have a tone control, the two circuits use similar clipping stages and the same maximum gain. But they differ in the filtering after the clipping section and in the type of clipping diodes: germanium in the DPL and silicon in the RAT, with the first type determining a softer clipping. Looking at the results we noticed how 72% of the times the DPL is classified as RAT, the Gain control was set to 1.0. On the other hand 60% of the times the RAT was classified as DPL, the Gain was set to 0.5 and the Tone (a first order lowpass) to 0 (no high frequency attenuation). This might be related to the clipping diodes; a low gain with silicon diodes might be comparable to the softer clipping of germanium diodes.

It is also relevant to observe that 808, TS9, MGS, and SD1, despite being based on the same circuit, with similar clipping sections and tone controls, are almost never confused.

To analyze the performance of our classifiers trained on continuous datasets and tested on discrete ones, we refer to Fig. 2. It can be noticed how the errors are similar to the

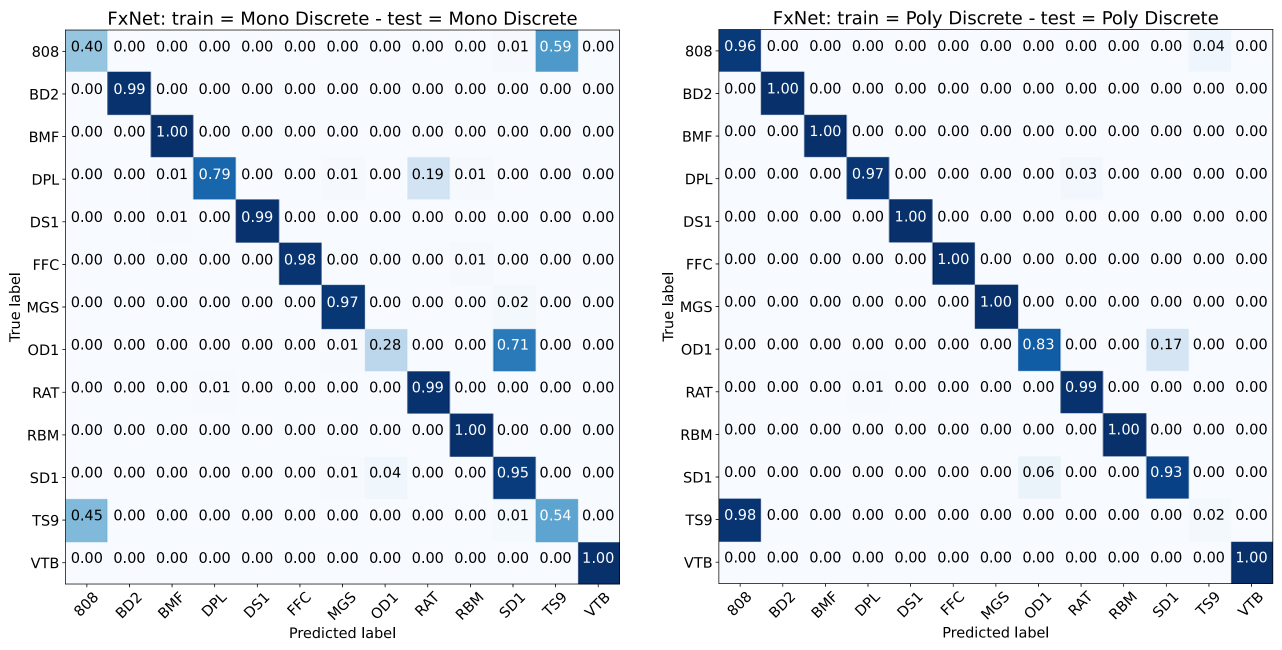

Fig. 1. Confusion matrices for discrete datasets.

previous cases, although a major impact on the performance is due to the misclassification of BD2. In this case we could not identify correlations between the different circuit designs and/or the controls values.

5.2 Settings Estimation

In this section we present the results for the settings estimation problem using the conditional network (SetNet-Cond) conditioned on the effect class ground truth. Table 5 shows the accuracy results, where all settings are estimated with an error below 0.1. Networks trained and tested on polyphonic datasets are the best performing, probably due to richer information content of the spectra with respect to monophonic samples. For both monophonic and polyphonic samples, the networks trained and tested on contin-

Table 5. SetNetCond accuracy (%).

	Test			
Train	Mono Disc.	Mono Cont.	Poly Disc.	Poly Cont.
Mono Disc.	80.06	68.56	···	···
Mono Cont.	68.51	85.14	···	···
Poly Disc.	···	···	90.75	75.74
Poly Cont.	···	···	88.93	97.01

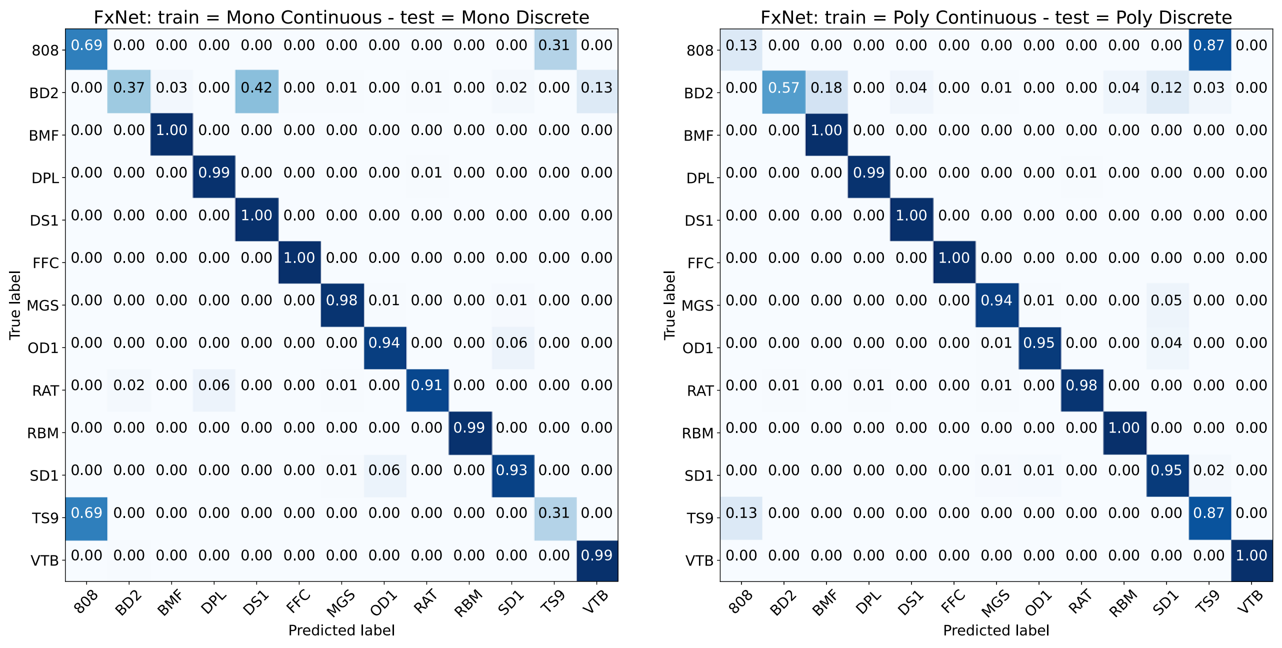

Fig. 2. Confusion matrices for test on discrete datasets.

Table 6. SetNetCond errors.

Train Set	Gain MAE *(RMSE)*	Tone/Eq MAE *(RMSE)*	Test Set
Mono Disc.	0.030 *(0.061)*	0.039 *(0.070)*	Mono Disc.
	0.064*(0.084)*	**0.044***(0.080)*	Mono Cont.
Mono Cont.	**0.062***(0.096)*	**0.067***(0.108)*	Mono Disc.
	0.033 *(0.045)*	0.039 *(0.072)*	Mono Cont.
Poly Disc.	**0.017***(0.033)*	**0.024***(0.047)*	Poly Cont.
	0.055 *(0.070)*	0.038 *(0.062)*	Poly Cont.
Poly Cont.	0.036 *(0.063)*	0.036 *(0.062)*	Poly Disc.
	0.020*(0.028)*	**0.019***(0.034)*	Poly Cont.
Avg	0.040 *(0.060)*	0.038 *(0.066)*	

uous settings reach higher accuracy than their counterparts trained and tested on discrete settings. This is somewhat surprising since the estimation of discrete values was expected to be a simpler problem to solve.

Further insights are offered by Table 6, where we show mean absolute error (MAE) and root mean square error (RMSE) for the different training and testing configurations. In the majority of cases (12 out of 16) the MAE is below 0.05 and for all cases except one the RMSE is below 0.1. We obtained the lowest errors when training and testing on polyphonic samples. The highest errors result from training on monophonic continuous and discrete samples and testing respectively on discrete and continuous ones. The table also includes the average errors for the Gain and Tone/Eq controls; the two are comparable, which shows that they present similar difficulty to the estimation. Fig. 3 shows the box-plots for the best and worst cases highlighted in Table 6.

Moreover we wanted to analyze the generalization capabilities of the networks trained on discrete settings as well as the performance of networks trained on continuous settings on discrete ones. Fig. 4 shows the scatter plots for the network trained on the Mono Discrete dataset when tested on the Mono Continuous dataset. For the Gain estimation we notice a bias toward the discrete values seen during training; also, the network manages to interpolate and estimate continuous values but seems to do it satisfactorily only for Gain values above 0.5. The Tone estimation seems to be less affected, and the output approximates fairly well the input uniform distribution. An explanation for this difference might reside in the fact that for the Tone control we chose a balanced set of discrete values ([0.0, 0.2, 0.5, 0.8, 1.0]). For the Gain control this was not possible. As explained in Sec. 2, most distortion effects do not produce any perceivable timbral difference for low gain values and some introduce attenuation.

The scatter plots for the network trained on the Mono Continuous and tested on the Mono Discrete datasets (Fig. 5) show some interesting behavior. Tested on a discrete dataset, the network fails to maintain the performance at the same levels as on the continuous one. In particular we notice a higher variance, especially for Gain = [0.2, 0.5] and most of the Tone values. Also, a skew in the estimations' distributions is visible.

To analyze these in more details, Figs. 6 and 7 show the mean error and skew as a function of Gain and Tone for those networks trained on discrete datasets and tested on continuous ones and vice-versa. Except for the case of Gain values below 0.1 in Fig. 7(a) (train on Poly Discrete and test on Poly Continuous), all mean errors for tests on opposite datasets are below 0.1. With the same exception, the mean errors for training on discrete datasets and test on continuous ones are lower than 0.05 (Figs. 6(a) and 7(a)). The skew for training on discrete and test on continuous datasets (Figs. 6(b) and 7(b)) is in many cases lower than the inverse conditions (Figs. 6(b) and 6(d) and 7(b) and 7(d)).

To conclude, even if the networks perform better on continuous datasets, there seems to be an argument for considering discrete values. A dataset that uses discrete values for the independent variables is easier to design, control, analyze, and extend or reduce. In our specific application an estimation error within 0.1 of the target value or some bias is acceptable. Also, the use of balanced values or a form of regularization in the cost function could help the interpolation in case of estimation on continuous values or unseen data.

5.3 Listening Test

To evaluate our observations about classification and estimation challenges, and their correlation with similarities and differences identified in the original analogue designs, we ran a listening test aimed at identifying perceptual differences between plugins. We used an AXY paradigm (also referred to as duo-trio) where participants were asked to identify which sample, X or Y, sounded the same as the reference A. A total of eight participants from our research group were involved. We did not record any anagraphic data or ask questions about previous experience but we can assume participants were not completely new to these kind of tests.

The test was split in three parts that compared 808 and TS9, OD1 and SD1, and DPL and RAT. Each part included 15 questions: 10 on the plugins under examination and five to validate the capability of each participant to identify clear perceptual differences. All samples were from the Mono Discrete dataset. The null hypothesis was that there is no perceptual difference between two samples. We analyzed each set of test results using a binomial test. Computing a one-tailed p-value and using a significance threshold of $\alpha = 0.05$ determined whether the null hypothesis was rejected.

Part 1 compared 808 and TS9. All the references were 808 samples that our model misclassified as TS9. Samples were randomly selected, obtaining a balance between note, guitar, pickup position, and settings values. In 9 out of the 10 cases, participants could not reliably distinguish between 808 and TS9 ($p_{one-tailed} = 0.297$). Therefore the null hypothesis could not be rejected and the two plugins might indeed be identical or very similar.

Part 2 compared OD1 and SD1. All the references were OD1 samples misclassified as SD1. Samples were randomly selected among those with Gain set to 0.2 or 0.5,

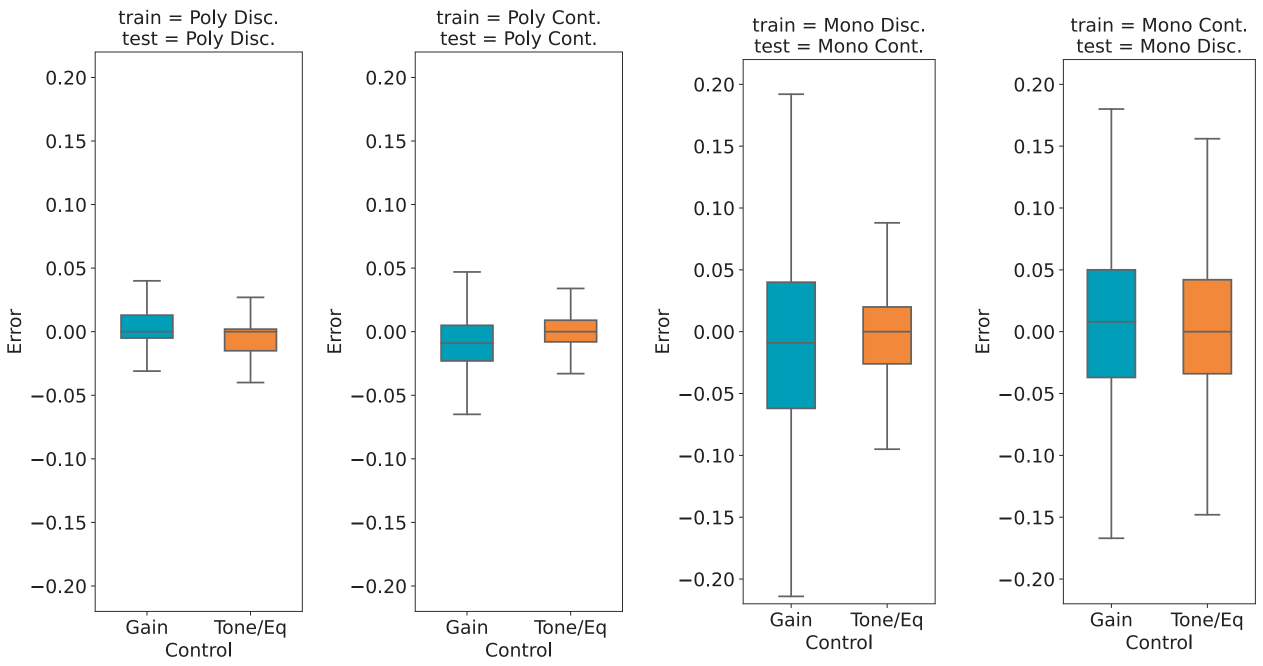

Fig. 3. Settings estimation errors (whiskers extend to 1.5 the interquartile range).

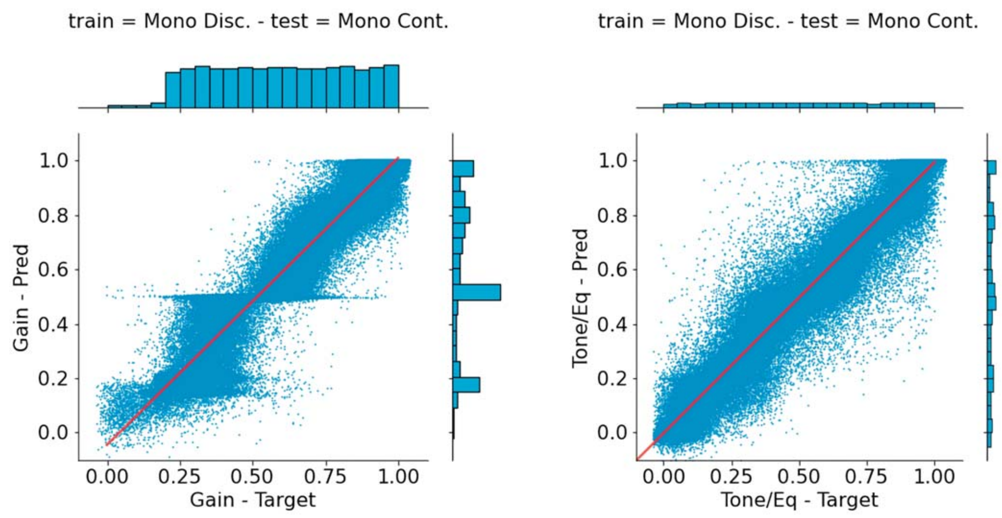

Fig. 4. Scatter plots for settings estimation on continuous datasets.

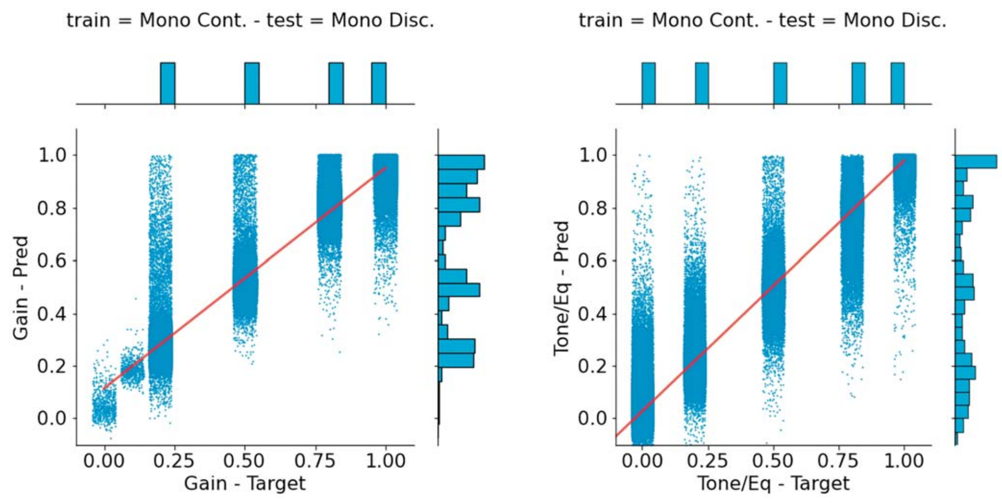

Fig. 5. Scatter plots for settings estimation on discrete datasets.

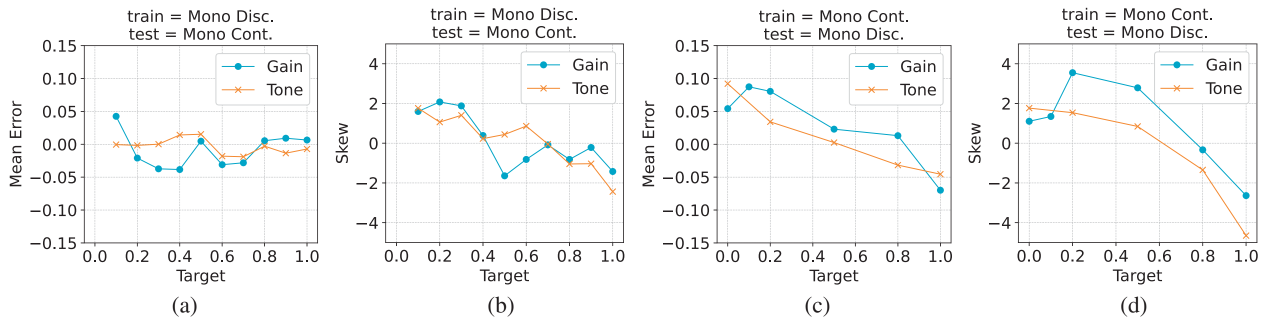

Fig. 6. Mean error and skew for networks trained/tested on monophonic samples.

being the most commonly misclassified ones. Also in this case, for 9 out of 10 conditions, participants could not reliably distinguish between OD1 and SD1 ($p_{one-tailed} = 0.292$).

Part 3 compared DPL and RAT. All the references were DPL samples misclassified as RAT. DPL samples were randomly selected among those with Gain set to 1.0 and compared with RAT samples with Gain set to 0.5 and Filter set to 0.0, being the specific settings most commonly misclassified. Here, for only 5 out of 10 conditions, participants could not reliably distinguish between DPL and RAT ($p_{one-tailed} = 0.13$).

In contrast, for all tests when the reference was compared against a sample where the classifier had 100% accuracy (e.g., 808 and BMF), all participants correctly identified the reference ($p_{one-tailed} = 0.0039$) and the null hypothesis could be rejected.

These results seem to confirm our observations (see Sec. 5.1), with misclassifications highlighting interesting commonalities between plugins and circuit designs. If a two-tailed test had been adopted the p-value would have doubled in almost every condition, making the results even stronger. Complete details about the test are available online (see Sec. 8).

6 DISCUSSION

Generally speaking, especially for digital distortion plugins, the amount of aliasing in the output signal could aid classification and estimation tasks. In our case plugins designed by *Audified* and *Mercuriall* (see Table 1) use 4x oversampling but we could not find details about *Analog Obsession*'s plugins. Also, we do not have details about the anti-aliasing lowpass filters adopted. Therefore we cannot exclude that our models might have used aliasing as a source of information, although one could argue that the amount of aliasing in the output is still a characteristic of each plugin. In this sense, both a human listener as well as an algorithm would interpret this as knowledge available for the task at hand. In any case, it would be interesting to devise some specific tests to evaluate the impact.

7 CONCLUSIONS AND FUTURE WORK

In this paper we introduced a CNN-based model for the classification of guitar effects and estimation of their parameters. Using electric guitar recordings of single notes, two-note intervals, and three or four-note chords, we were able to classify with high accuracy samples processed with 13 overdrive, distortion, and fuzz plugins. The plugins we used are emulations of some of the most famous and commonly used analogue guitar effect pedals. We were also able to estimate the parameters settings with high precision. For this work we generated a dataset of processed monophonic and polyphonic samples; the plugins' settings were either combinations of discrete values commonly used by musicians or continuous values randomly extracted from a uniform distribution. This allowed us to gain further understanding about the performance on datasets generated using discrete or continuous independent variables. We described the benefits of discrete datasets and postulated about the possibility of obtaining equally high performances.

To the best of our knowledge our work is the first attempt at both classification and parameter estimation of specific

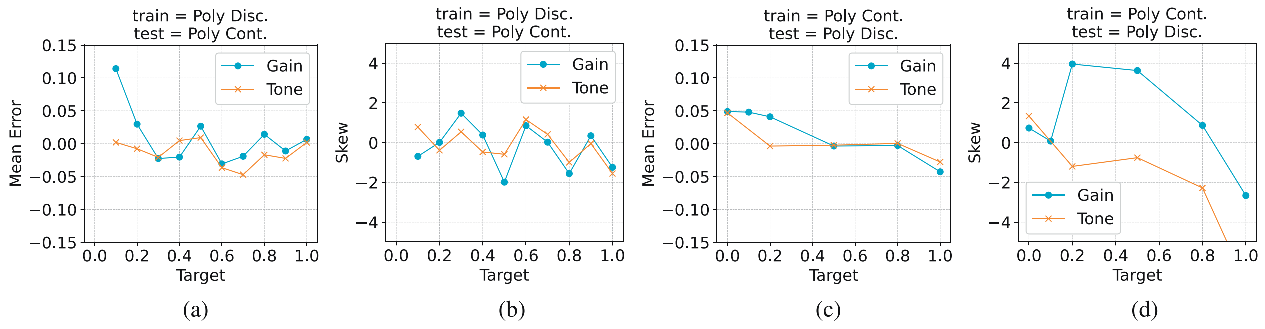

Fig. 7. Mean error and skew for networks trained/tested on polyphonic samples.

guitar effect units. Differently from previous literature, conditioning on the effect class allows for settings' inference without the need for different input representations, network architectures, or separate training.

Being able to extrapolate information about the specific signal chain used in a recording could benefit music production; artists, engineers, and producers often rely on "reference" sounds from other recordings. This knowledge could also be applied to automatic music transcription [8], music education [10], or musicology [11]. Musicians, genres, and styles are identified by and associated with specific sounds and effects; this could be applied to intelligent music search and recommendation systems.

Future work might branch out in different directions: our model could be compared with human performance; the architecture and datasets could be extended for higher accuracy and better generalization on unseen data or study the problem with limited training data [40]. The research could be extended to other nonlinear units, different classes of effects, or tested on other guitars or instruments. Also of interest would be the case of audio from real guitar pedals instead of digital emulations.

8 DATA AND CODE AVAILABILITY

Code:
- https://doi.org/10.5281/zenodo.4670359

Models:
- https://doi.org/10.5281/zenodo.4670384

Dataset:
- Mono Disc. - https://doi.org/10.5281/zenodo.4298000
- Mono Cont. - https://doi.org/10.5281/zenodo.4296040
- Poly Disc. - https://doi.org/10.5281/zenodo.4298025
- Poly Cont. - https://doi.org/10.5281/zenodo.4298017

Extended results:
- https://mcomunita.github.io/gfx_classifier_page

9 ACKNOWLEDGMENT

This study has been funded by UKRI and EPSRC as part of the "UKRI Centre for Doctoral Training in Artificial Intelligence and Music," under grant EP/S022694/1.

10 REFERENCES

[1] A. Case, "Recording Electric Guitar—The Science and the Myth," *J. Audio Eng. Soc.*, vol. 58, no. 1/2, pp. 80–83 (2010 Jan.).

[2] J. Blair, *Southern California Surf Music, 1960–1966 (Images of America)* (Arcadia Publishing, Mount Pleasant, SC, 2015).

[3] S. Williams, "Tubby's Dub Style: The Live Art of Record Production," in S. Frith and S. Zagorski-Thomas (Eds.), *The Art of Record Production: An Introductory Reader for a New Academic Field*, pp. 235–246 (Ashgate, Farnham, UK, 2012).

[4] P. Prown and L. Sharken, *Gear Secrets of the Guitar Legends: How to Sound Like Your Favorite Players* (Backbeat Books, London, UK, 2003).

[5] U. Zölzer, *DAFX: Digital Audio Effects* (John Wiley & Sons, Hoboken, NJ, 2011).

[6] J. Pakarinen and D. T. Yeh, "A Review of Digital Techniques for Modeling Vacuum-Tube Guitar Amplifiers," *Comp. Music J.*, vol. 33, no. 2, pp. 85–100 (2009).

[7] P. Herrera-Boyer, G. Peeters, and S. Dubnov, "Automatic Classification of Musical Instrument Sounds," *J. New Music Res.*, vol. 32, no. 1, pp. 3–21 (2003 Mar.).

[8] E. Benetos, S. Dixon, D. Giannoulis, H. Kirchhoff, and A. Klapuri, "Automatic Music Transcription: Challenges and Future Directions," *J. Int. Inf. Syst.*, vol. 41, no. 3, pp. 407–434 (2013).

[9] 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 *Proceedings of the 17th International Conference on Digital Audio Effects (DAFx-14)*, pp. 219–226 (Erlangen, Germany) (2014 Sep.).

[10] C. Dittmar, E. Cano, J. Abeßer, and S. Grollmisch, "Music Information Retrieval Meets Music Education," in *Multimodal Music Processing, Dagstuhl FollowUps*, pp. 95–119, vol. 3 (2012).

[11] J. Abeßer, K. Frieler, E. Cano, M. Pfleiderer, and W.-G. Zaddach, "Score-Informed Analysis of Tuning, Intonation, Pitch Modulation, and Dynamics in Jazz Solos," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 25, no. 1, pp. 168–177 (2017 Jan.).

[12] I. Barbancho, L. J. Tardon, A. M. Barbancho, and S. Sammartino, "Pitch and Played String Estimation in Classic and Acoustic Guitars," presented at the *126th Convention of the Audio Engineering Society* (2009 May), paper 7701.

[13] J. Abeßer, "Automatic String Detection for Bass Guitar and Electric Guitar," in *Proceedings of the International Symposium on Computer Music Modeling and Retrieval*, pp. 333–352 (London, UK) (2012 Jun.).

[14] C. Dittmar, A. Männchen, and J. Abeßer, "Real-Time Guitar String Detection for Music Education Software," in *Proceedings of the 14th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS)*, pp. 1–4 (Paris, France) (2013 Jul.).

[15] T. Geib, M. Schmitt, and B. Schuller, "Automatic Guitar String Detection by String-Inverse Frequency Estimation," in *Proceedings of the INFORMATIK*, pp. 127–138 (Chemnitz, Germany) (2017 Sep.).

[16] J. M. Hjerrild and M. G. Christensen, "Estimation of Guitar String, Fret and Plucking Position Using Parametric Pitch Estimation," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 151–155 (Brighton, UK) (2019 May).

[17] J. M. Hjerrild, S. Willemsen, and M. G. Christensen, "Physical Models for Fast Estimation of Guitar String, Fret and Plucking Position," in *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pp. 155–159 (New Paltz, NY) (2019 Oct.).

[18] Z. Mohamad, S. Dixon, and C. Harte, "Pickup Position and Plucking Point Estimation on an Electric Guitar via Autocorrelation," *J. Acoust. Soc. Am.*, vol. 142, no. 6, pp. 3530–3540 (2017 Dec.).

[19] Z. Mohamad, S. Dixon, and C. Harte, "Pickup Position and Plucking Point Estimation on an Electric Guitar," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 651–655 (New Orleans, LA) (2017 Mar.).

[20] J. Abeßer, H. Lukashevich, and G. Schuller, "Feature-Based Extraction of Plucking and Expression Styles of the Electric Bass Guitar," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 2290–2293 (Dallas, TX) (2010 Mar.).

[21] G. Schuller, J. Abeßer, and C. Kehling, "Parameter Extraction for Bass Guitar Sound Models Including Playing Styles," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 404–408 (South Brisbane, QLD) (2015 Apr.).

[22] L. Su, L.-F. Yu, and Y.-H. Yang, "Sparse Cepstral and Phase Codes for Guitar Playing Technique Classification," in *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)International Society for Music Information Retrieval Conference (ISMIR)*, pp. 9–14 (Taipei, Taiwan) (2014 Oct.).

[23] C.-Y. Wang, P.-C. Chang, J.-J. Ding, T.-C. Tai, A. Santoso, Y.-T. Liu, and J.-C. Wang, "Spectral-Temporal Receptive Field-Based Descriptors and Hierarchical Cascade Deep Belief Network for Guitar Playing Technique Classification," *IEEE Trans. Cyber.*, pp. 1–12 (2020 Sep.).

[24] R. Foulon, P. Roy, and F. Pachet, "Automatic Classification of Guitar Playing Modes," in *Proceedings of the International Symposium on Computer Music Multidisciplinary Research*, pp. 58–71 (Marseille, France) (2013 Oct.).

[25] A. M. Barbancho, A. Klapuri, L. J. Tardón, and I. Barbancho, "Automatic Transcription of Guitar Chords and Fingering From Audio," *IEEE Trans. Audio Speech Lang. Process.*, vol. 20, no. 3, pp. 915–921 (2012 Mar.).

[26] K. Dosenbach, W. Fohl, and A. Meisel, "Identification of Individual Guitar Sounds by Support Vector Machines," in *Proceedings of the 11th International Conference on Digital Audio Effects (DAFx-08)*, pp. 317-320 (Espoo, Finland) (2008 Sep.).

[27] D. Johnson and G. Tzanetakis, "Guitar Model Recognition From Single Instrument Audio Recordings," in *Proceedings of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, pp. 370–375 (Victoria, BC) (2015 Aug.).

[28] R. d. C. R. Profeta and G. Schuller, "Feature-Based Classification of Electric Guitar Types," in P. Cellier and K. Driessens (Eds.), *Machine Learning and Knowledge Discovery in Databases*, pp. 478–484 (Springer, New York, NY, 2020).

[29] R. Profeta and G. Schuller, "Comparison of Human and Machine Recognition of Electric Guitar Types," presented at the *147th Convention of the Audio Engineering Society* (2019 Oct.), paper 10315.

[30] M. Stein, J. Abeßer, C. Dittmar, and G. Schuller, "Automatic Detection of Audio Effects in Guitar and Bass Recordings," presented at the *128th Convention of the Audio Engineering Society* (2010 May), paper 8013.

[31] M. Stein, "Automatic Detection of Multiple, Cascaded Audio Effects in Guitar Recordings," in *Proceedings of the 13th International Conference on Digital Audio Effects (DAFx-10)*, pp. 4–7 (Graz, Austria) (2010 Sep.).

[32] F. Eichas, M. Fink, and U. Zölzer, "Feature Design for the Classification of Audio Effect Units by Input/Output Measurements," in *Proceedings of the 18th International Conference on Digital Audio Effects ((DAFx-15)DAFx-15)* (Trondheim, Norway) (2015 Nov./Dec.).

[33] M. Schmitt and B. Schuller, "Recognising Guitar Effects - Which Acoustic Features Really Matter?" in *Proceedings of the INFORMATIK* (Chemnitz, Germany) (2017 Sep.).

[34] H. Jürgens, R. Hinrichs, and J. Ostermann, "Recognizing Guitar Effects and Their Parameter Settings," in *Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx2020)* (Vienna, Austria) (2020 Sep.).

[35] F. Eichas and U. Zölzer, "Gray-Box Modeling of Guitar Amplifiers," *J. Audio Eng. Soc.*, vol. 66, no. 12, pp. 1006–1015 (2018 Dec.).

[36] B. McFee, C. Raffel, D. Liang, D. P. W. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and Music Signal Analysis in Python," in *Proceedings of the 14th Python in Science Conference*, vol. 8, pp. 18–24 (2015 Jan.).

[37] J. Abeßer, "A Review of Deep Learning Based Methods for Acoustic Scene Classification," *Appl. Sci.*, vol. 10, no. 6 (2020 Mar.). https://doi.org/10.3390/app10062020.

[38] S. Ruder, "An Overview of Multi-Task Learning in Deep Neural Networks," *arXiv preprint arXiv:1706.05098* (2017).

[39] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv preprint arXiv:1412.6980* (2014).

[40] C. J. Steinmetz and J. D. Reiss, "Efficient Neural Networks for Real-Time Analog Audio Effect Modeling," *arXiv preprint arXiv:2102.06200* (2021).

THE AUTHORS

Marco Comunità

Dan Stowell

Joshua D. Reiss

Marco Comunità is a PhD candidate with the Centre for Digital Music at Queen Mary University of London. He also collaborates with the Audio Experience Design Lab at Imperial College London. His research activity includes machine learning and deep learning applied to audio signal processing, neural synthesis of audio effects, 3D binaural sound rendering, and headphones acoustics. Marco holds a BEng in Computer Engineering and an MEng in Electronic Engineering from Sapienza University of Rome and an MSc in Sound and Music Computing from Queen Mary University of London.

•

Dan Stowell is Associate Professor of AI & Biodiversity at Tilburg University and Naturalis Biodiversity Centre. His work centers on using machine learning to explore the mysteries of sound, with a specific focus on animal sound and environmental soundscapes. He co-founded the Machine Listening Lab at Queen Mary University of London and is a Turing Fellow at the UK's Alan Turing Institute. In 2015 Dan co-founded the spin-out company to produce the app Warblr, which recognizes bird sounds automatically—used regularly by many thousands of people around the UK.

•

Josh Reiss is Professor of Audio Engineering with the Centre for Digital Music at Queen Mary University of London. He has published more than 200 scientific papers (including over 50 in premiere journals and 6 best paper awards) and co-authored two books. His research has been featured in dozens of original articles and interviews on TV and radio and in the press. He is a Fellow and President-Elect of the Audio Engineering Society (AES) and chair of their Publications Policy Committee. He co-founded the highly successful spin-out company, LandR, and recently co-founded Tonz and Nemisindo, also based on his team's research. He maintains a popular blog, YouTube channel, and Twitter feed for scientific education and dissemination of research activities.