RECOGNIZING GUITAR EFFECTS AND THEIR PARAMETER SETTINGS

Henrik Jürgens, Reemt Hinrichs and Jörn Ostermann

Institut für Informationsverarbeitung
Leibniz Universität Hannover
Hannover, Germany
juergens@tnt.uni-hannover.de

ABSTRACT

Guitar effects are commonly used in popular music to shape the guitar sound to fit specific genres or to create more variety within musical compositions. The sound is not only determined by the choice of the guitar effect, but also heavily depends on the parameter settings of the effect. This paper introduces a method to estimate the parameter settings of guitar effects, which makes it possible to reconstruct the effect and its settings from an audio recording of a guitar. The method utilizes audio feature extraction and shallow neural networks, which are trained on data created specifically for this task. The results show that the method is generally suited for this task with average estimation errors of $\pm 5\% - \pm 16\%$ of different parameter scales and could potentially perform near the level of a human expert.

1. INTRODUCTION

Extracting information from audio becomes continually more important. Music needs to be classified and categorized not only for streaming services, but also in the music production itself. Music information retrieval becomes more viable with increasing processing power on consumer devices. This brings automatic mixing and more complex effect chains closer to reality [1]. The electric guitar is a widely used instrument in modern music and providing methods to retrieve information from guitar tracks can be a great help. Effects from recorded songs could be reconstructed quickly and provide a better workflow, when searching for the right guitar sound. Streaming services could not only create playlists based on current metadata, but also based on the specific sound of the guitars. Especially combined with more advanced deep learning based source separation, such as Spleeter [2] and Open-Unmix [3], analyzing individual instruments from recorded and mixed songs may become viable. Lastly, an application in music education might be possible, training novice guitar players to choose their effects and settings, similarly to concepts in [4], helping them to develop their unique sound. Even though skilled guitar players can manually reconstruct the effects and their parameter settings from audio, this is not an option for analyzing the massive amount of music and different musical styles, that are published today.

The sound of the electric guitar is largely characterized by the used effects and their parameter settings, transforming a sound from very harsh distortion, as used in metal music, to spacious, atmospheric, modulated sounds of psychedelic rock. Digital implementations of the most common guitar effects can be found in [5].

In this work we present a method to estimate the parameter settings of guitar effects. In Section 2 previous work related to the topic is summarized. Section 3 gives a short overview of the method for effect classification and parameter settings estimation presented in this paper. Section 4 describes our reimplementation of the effect classification. Section 5 our proposed method for the estimation of parameter settings is explained and then evaluated in Section 6. The Sections 7 and 8 present future work and summarize the results of this work.

2. PREVIOUS WORK

Before methods for the recognition of guitar effects were published, the automated distinction of different guitars and different instruments was examined in [6] and [7]. Stein et al. [8] investigated the classification of eleven different digital guitar effect classes. For this publication they assembled a database of guitar samples, which were processed with different effects. This database was also used as training and evaluation set for this paper. Stein et al. achieve a classification accuracy of 95.5 %, meaning that 95.5 % of the samples in the test set have been assigned the correct effect class. Similar to the previously mentioned publications, a Support Vector Machine (SVM) was used as classifier. Stein expanded his approach in [9] to classify multiple cascaded guitar effects.

Schmitt and Schuller [10] presented a more detailed analysis of the guitar effect classification, investigating the feature importance. Their approach is similar to Stein et al. [8], but has a less complex feature extraction stage. Out of all their experiments the maximum classification accuracy of the monophonic samples in the database is 97.8 %. So even though a direct comparison to Stein et al. for all samples is not possible, their classification accuracy is in a similar range.

Yee-King et al. [11] provided an interesting method for reconstructing the parameter settings of a synthesizer from an audio sample. They compared more recent methods with learned features, which are often used in speech recognition, such as neural networks with Long Short-Term Memory (LSTM) cells.

In [12] Sheng et al. proposed a method to extract parameters of a dynamic range compressor from a reference sound. They did solve the problem as a regression problem using random forest regression and linear regression. They expanded their approach in [13] by using deep neural networks to learn features instead of using handcrafted features.

To the best of our knowledge, there are currently no publications providing methods for estimating the effect parameter settings of guitar effects.

3. OVERVIEW

The approach that is proposed in this paper for extracting the guitar effect and its parameter settings is displayed in Figure 1. First, the effect class needs to be determined. Afterwards, the effect parameters can be estimated for this specific effect. This approach allows to design specific algorithms for each effect class.

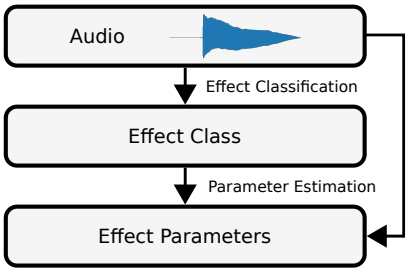

Figure 1: *Pipeline for determining the guitar effect and its settings*

4. GUITAR EFFECT RECOGNITION

The ten guitar effects of the database [8] can be split into the three categories: Nonlinear effects, ambience effects and modulation effects. Additionally, there is one extra class which contains the unprocessed samples and samples that have been processed with an equalizer to further diversify the sounds. Thus, all eleven classes contain the same amount of processed guitar samples. A list of all effects and their categories can be found in Table 1.

Table 1: *List of effect classes in the database and their category*

Category	**Effect Classes**
Nonlinear	Distortion, Overdrive
Ambience	Feedback Delay, Slapback Delay, Reverb
Modulation	Chorus, Flanger, Phaser, Tremolo, Vibrato
Clean	Unprocessed & Equalized

There are 624 monophonic samples, containing all pitches of the guitar until the 12th fret in standard tuning, recorded by using two different guitars and two different pickup settings each. Additionally, 420 polyphonic samples are part of the database, covering various intervals and chords spread over the guitar neck, also using the same two guitars and two pickup settings. These 1044 samples have been processed with 3 parameter settings of every effect, resulting in roughly 16 hours (excluding silence) of 2 second clips or about 34,500 samples respectively. To avoid bias, the guitar samples need to be peak normalized before feature extraction, since the audio level could allow the classifier to draw a conclusion on the effect class.

For training as well as the prediction of new samples, four steps are incorporated, as shown in Figure 2. The silence is cut from the beginning of the samples. This is achieved using onset detection to find the start of the played note. With the Librosa [14] implementation of the onset detection, only the strongest onset must be detected. This can be achieved by setting the pre max

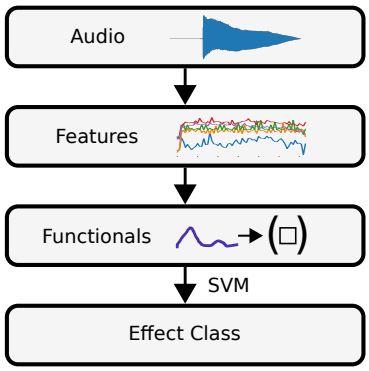

Figure 2: *Processing pipeline for recognizing guitar effects*

and post max values to 20000 samples each. This is especially important since some of the polyphonic samples do have less than half a second of silence in the beginning. Several features are then extracted. A complete list of the used audio features can be found in Table 2.

The frameworks Librosa [14] and a Python interface for Praat [15] have been used for feature extraction. All features that are extracted with Librosa use a Hann window with a length of 2048 samples and 25% overlap. The features extracted with Praat use a window length of 2352 samples (four periods of the lowest frequency 75 Hz) at a sample rate of 44.1 kHz. For each window, four values are calculated, oversampling the resulting time series by a factor of four. The harmonics to noise ratio is calculated only once for the whole guitar sample. Besides the time series, deltas, local estimates of the derivatives, of those time series are used, a combination of which has proven itself very useful in the work of Schmitt and Schuller [10]. Beyond the features from Praat and Librosa, an additional feature is extracted, which tracks the unwrapped phase of the maximum frequency bin of a short-time Fourier transform. In line with the results of Schmitt and Schuller, we have found that modulation effects, especially the phaser, are the hardest to classify. Thus, we implemented a new feature to detect modulations of the phase, assuming that the unwrapped phase of the maximum frequency bin is approximately linear if the sample is not processed with modulation effects.

Every extracted feature is then processed with several functionals, mappings from time series to scalars, to obtain scalar values, which can be consolidated in a vector for the SVM from the framework Scikit-learn [16]. This allows classification independent of the length of the input signal under the condition that the input signal only contains one onset, since more onsets radically change those functionals tracking the gradients of audio features. All used functionals can be found in Table 3. For the Fast Fourier Transform (FFT)-Functional, the audio feature time series is filled up with zeros to a length of 1024. The maximum of the interpolated spectrum is then used to estimate whether an audio feature is periodic and to what extent. Modulation effects generally lead to periodic audio features, since they modulate either pitch, phase, amplitude or a combination of those periodically.

With 12 functionals and 54 time series audio features and one scalar audio feature, the input vector for the SVM has a size of $12 \cdot 54 + 1 = 649$. Since we used less functionals and features than Schmitt and Schuller, the input vector of our SVM is only a tenth of the size, resulting in much faster training times, since the

Table 2: *Used audio features for guitar effect recognition*

Feature # Features / Frame	**Framework**	**Time Series**	**Delta**
Mel Frequency Cepstral Coefficients 20 + 20 Δ	Librosa	Yes	Yes
Spectral Contrast 7	Librosa	Yes	No
Zero Crossing Rate 1 + 1 Δ	Librosa	Yes	Yes
Root Mean Square Energy 1 + 1 Δ	Librosa	Yes	Yes
Unwrapped Phase of Maximum Frequency Bin 1	(None)	Yes	No
Pitch Curve 1	Praat	Yes	No
Voiced Probability 1	Praat	Yes	No
Harmonics to Noise Ratio -	Praat	No	No

Table 3: *Used functionals and number of scalar values as output of the functionals*

Functional	**# Scalars**
Maximum	1
Minimum	1
Average	1
Standard Deviation	1
2 Linear Regression Coefficients + Residual	3
3 Quadratic Regression Coefficients + Residual	4
Maximum of FFT	1
Sum	12

training time for SVM scales at least quadratically with the input vector size [17]. This allows for more flexibility through shorter testing cycles for new features. Despite the smaller input vector, similar results were achieved, classifying the guitar samples with an accuracy of 94.85 %.

We have also investigated the feature importance using an extra trees classifier [18] to determine the most important scalars within the 649-dimensional input vector for the SVM. In another experiment, we excluded each feature and each functional to train and evaluate again and then measure the accuracy difference. In those evaluations, the lower Mel Frequency Cepstral Coefficients (MFCC), Root Mean Square (RMS) and Spectral Contrast and their deltas were the most important features and the FFT, max, min, standard deviation and linear regression were the most important functionals. This provided a good insight for selecting the features for the estimation of effect settings.

5. ESTIMATION OF EFFECT SETTINGS

We first attempted to apply our reimplementation of the approach of [10] to the parameter estimation. Therefore, the gain parameter of the distortion effect was quantized into five classes, each covering an interval of 0.2 of the parameter range from 0.0 to 1.0. We then applied our reimplementation of the effect classification to classify the intervals of the parameter range. This way a classification accuracy of 66% was reached. Despite that, when adding the error, that results from quantizing the parameter range into classes, onto the estimation, the overall result was of similar accuracy as assuming random settings. Therefore it was necessary to develop a new method for the effect parameter estimation.

When developing the method for estimating the effect settings, we first investigated whether it is possible to separate effect settings by a certain feature. For this investigation, we chose the distortion effect with the three settings that were already included in the database. We were able to classify those three settings with a linear SVM using only the two linear regression coefficients of three MFCC with 98 % accuracy. Based on these encouraging results we decided to investigate whether a more precise parameter estimation was possible. Since the problem of estimating continuously adjustable effect parameters is more a regression problem than a classification problem, we used neural networks instead of SVMs. Neural networks are well suited to solve regression problems if a mean square loss function and linear output nodes are used [19].

To train the neural network using Keras [20], new samples had to be generated which represent the full range of the parameters. We used the digital audio workstation Reaper and the unprocessed samples from the database supplied by Stein et al. [8]. This way, we could process the samples with uniform randomly distributed settings of the distortion effect parameters, namely tone, edge and gain. Tone determines the cutoff frequency of a low pass filter; edge determines the gain of resonant peak at the cutoff frequency. After that the signal is multiplied by the gain factor and distorted by the nonlinearity. The processed samples are 624 monophonic and 420 polyphonic samples. Using a Reaper script about 1000 samples can be generated in one hour on a typical desktop computer.

The neural network for parameter estimation can neither have many layers nor many neurons per layer, since the number of trained parameters should not be greater than the number of training samples. Therefore, we started out using a neural network with only one hidden layer varying the size in powers of two. The best results could be achieved using 32 neurons in the hidden layer. We also tested using more layers, but this did not yield better results than using the previously described configuration.

Additionally, using a batch norm layer in front of the hidden layer appears to prevent overfitting, since the input vectors are normalized differently depending on the batch, providing more variety. Thus, the training can span over 1000 epochs with a monotonically decreasing validation error. For our evaluation we excluded 30% of the data as test set and used the remain 70 % for training and validation. The neural network has rectified linear activation functions and is trained using an Adam optimizer with a learning rate of 0.01, mean squared error as loss function, a batch size of 32 and a 5-fold cross-validation on the training data.

The previously described implemented pipeline is depicted in Figure 3. After developing the parameter estimation for the distortion effect, we investigated how the same approach would perform

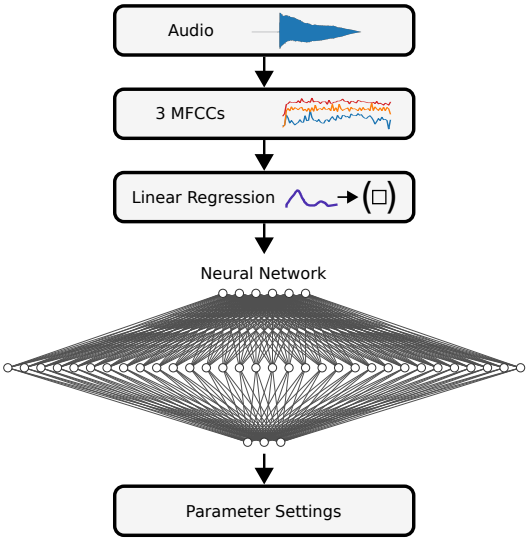

Figure 3: *Pipeline for the estimation of the three parameters of the distortion effect*

on two further effects. We chose the delay with the parameters delay length and volume of the wet signal and the tremolo effect with the parameters modulation frequency and modulation depth. The delay time ranges from 0 ms to 1000 ms, the tremolo frequency ranges from 0.1 Hz to 30 Hz. By choosing these effects, we implemented the method for one nonlinear time-invariant, one linear time-invariant and one linear time-variant effect. We have used different features and functionals that are better suited for the specific effect parameters, but the approach remains the same.

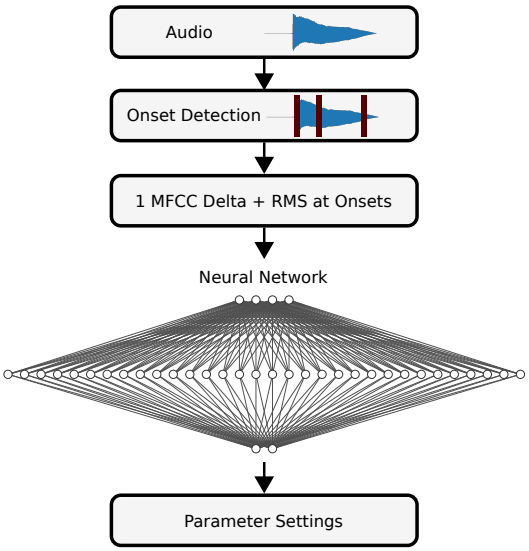

Figure 4: *Pipeline for estimation of the three parameters of the delay effect*

For the delay, we used the onset detection not only for cutting the silence in the beginning but also to try to find the delayed onset. Since the feedback was disabled only one onset had to be found. Despite that, extracting features at two onsets has proven to be more reliable, since the wrong onset can be detected and extracting features at the second onset provides redundancy for this case. Finding the correct onset is not trivial because the second onset is superimposed on the original signal with arbitrary phase relation. Out of different combinations of RMS-energy, MFCCs, spectral flow (onset strength) and their respective deltas, using the first MFCC delta and the RMS at the onset position yielded the best results. For the MFCC delta, not only the frame at the onset is used, but rather a sum of MFCC delta of the five frames before the detected onset. So, both features provide a measure of signal energy at the onset, which is increasing with the onset of the delayed signal.

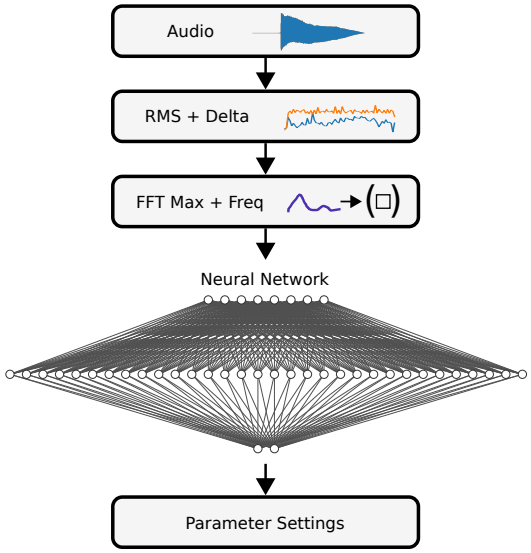

Figure 5: *Pipeline for estimation of the three parameters of the tremolo effect*

The tremolo effect modulates the volume of the incoming signal periodically. Consequently, we try to analyze the volume of the signal using the RMS and its delta. Next, the FFT of those time series is calculated to find periodic modulation. To provide sufficient resolution, the FFT is interpolated by appending zeros until a size of 1024 is reached. This also ensures the corresponding frequencies of the frequency bins stay the same, independent of the length of the input signal. After identifying the maximum of the FFT spectrum, this maximum and 64 bins around it are set to zero and a second maximum is identified. This way, a redundancy is provided, in case the first maximum is not caused by the tremolo effect, but rather by the natural decay of the guitar signal, which generally causes a peak at around 1 Hz, making it harder to identify low modulation frequencies. The amplitude of the maximums and their bin numbers are then standardized and processed by the neural network.

6. EVALUATION

We used the absolute error e from the ground truth parameter setting s_{true}, a continuous value from zero to one, to the estimated setting s_{est} to measure the performance.

$$e = |s_{true} - s_{est}|$$

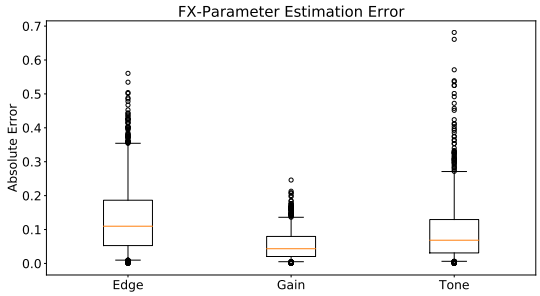

Figure 6: *Absolute error of the estimation of the distortion effect parameters: the median is marked by an orange line, the surrounding box covers data from the 25% quartile to the 75% quartile. The whiskers cover the 5th percentile to the 95th percentile; everything outside of those percentiles is an outlier, displayed as circle.*

Figure 6 displays the absolute error for the parameters edge, gain and tone of the distortion effect. The gain parameter is predicted with the smallest error of 0.05 on average. This parameter also generally has the largest subjective impact on the sound. The parameters edge and tone can be estimated with an average absolute error of 0.14 and 0.10 respectively. The extreme errors are also much greater than those of the gain parameter.

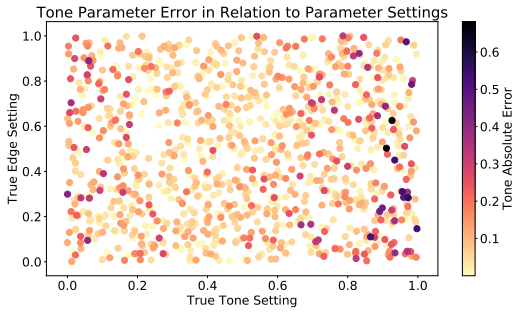

Figure 7: *Absolute error of the tone parameter estimation plotted against the ground truth setting of tone and edge*

Figure 7 gives more insight into the outliers, which are depicted as the darkest points. Especially with high tone parameter settings, corresponding to high cutoff frequency, changes are hard to detect, since guitar signals have their highest amplitudes in the frequency spectrum in the first harmonics. Thus, the MFCC will not change very much when only the high frequency content is affected by changing the tone parameter. The outliers of the edge parameter show no such clear trends. One should consider that the edge parameter has a much smaller effect on the sound as it is perceived by a human than the other two parameters. So, errors in the setting of this parameter will not be noticed as clearly.

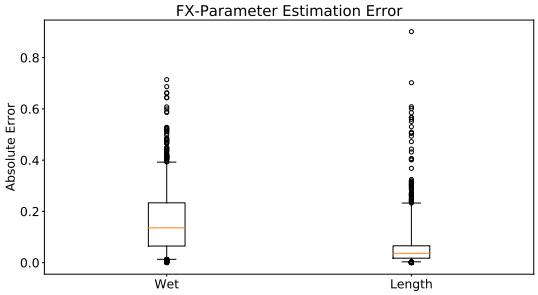

Figure 8: *Absolute error of the delay effect parameters, boxplot properties are as described in Figure 6*

The delay parameters wet and length can be estimated with an average absolute error of 0.16 and 0.06 respectively. Figure 8 shows that the extreme errors for both parameters are above 0.7. The wet parameter estimation performs worse than the length, since errors in the length parameter estimation will lead to errors in the wet parameter estimation, since the wrong segment of the audio sample will be analyzed.

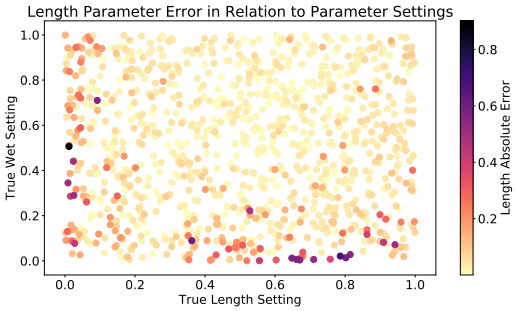

Figure 9: *Absolute error of the length parameter estimation plotted against the ground truth setting of length and wet*

Figure 9 shows that these outliers are all at either low wet settings, where the delay will be inaudible or at low length settings. A length setting below 0.05 corresponds to a delay below 50 ms, where the delay rather acts as a comb filter. Thus, most of the outliers are negligible, since they are results of settings that would not be chosen, when a delay effect is desired. Analyzing the wet parameter error, we observed the same phenomena, with the difference of an overall greater error.

The tremolo parameters depth and frequency can be estimated with an average absolute error of 0.08 and 0.06 respectively. Figure 10 also shows some outliers, but the median error here is much lower compared to other effects. The depth error is influenced by the frequency error, since if the wrong FFT bin is chosen, the magnitude of the wrong bin will be analyzed.

Figure 11 shows that very low depth settings, leading to the effect being inaudible, make the frequency estimation harder. Low frequency settings can be indistinguishable from the natural decay

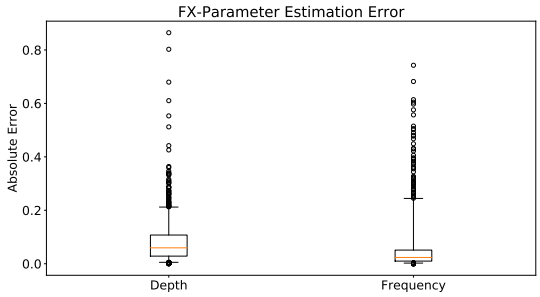

Figure 10: *Absolute error of the estimation of the tremolo effect parameters, boxplot properties are as described in Figure 6*

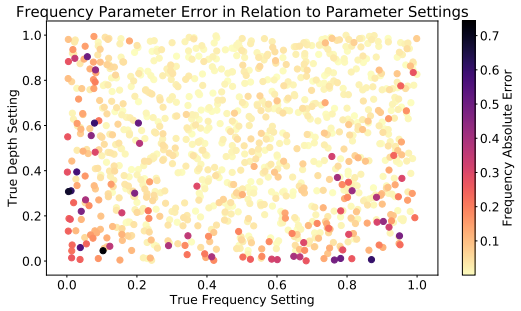

Figure 11: *Absolute error of the frequency parameter estimation plotted against the ground truth setting of frequency and depth*

of the guitar string. Especially if the modulation curve decreases while the sample is decaying, the effect can be inaudible. The greatest outliers of the depth parameter error seem to stem from the same phenomenon. As with the delay, it can be assumed that the settings producing the greatest errors, are being used the least in typical guitar recordings.

From our own experience of creating guitar sounds, we estimate that a guitar player can be sufficiently accurate by only setting effect parameters in steps of 0.1, which would result in an absolute error of 0.05. This would mean that the average errors of the proposed method are close to the precision of a human expert, especially considering, that currently only a single note is analyzed. If a whole song or song section was analyzed, the results could potentially be significantly improved by taking an average of the estimated settings for each onset.

7. FUTURE WORK

Future research should investigate whether better results could be reached using architectures with learned features for this problem, such as convolutional neural networks or neural networks with long short-term memory cells as suggested by Humphrey et al. [21] and similar to the implementations of Sheng et al. in [13]. It might be effective to utilize transfer learning using networks that have previously been trained for phoneme recognition, since phoneme recognition implicitly determines filter parameters (such as jaw, tongue and lip positions) of the human voice, necessary to distinguish phonemes independent of the speaker. Additionally, it might be useful to investigate how well humans can estimate individual effect parameters to evaluate if the current method's performance is sufficient for the designated use case. This could be done in a survey by either having humans reproduce effect settings from audio or humans rating differences in effect settings in a MUSHRA test.

8. CONCLUSION

This paper presents a novel method for estimating guitar effect settings. For this purpose, specific features catered to the characteristics of each effect class have been selected. These features are then processed by shallow neural networks to perform a mapping from the feature data to the estimated parameter values. This method can estimate effect parameters with errors potentially close to those of human experts. The method has been implemented and tested successfully for a distortion, delay and tremolo effect, representing one effect from each major guitar effect category. The average errors for the different parameters are between $\pm 5\%$ and $\pm 16\%$. The estimation is most accurate for parameters which have the greatest impact on the sound and commonly used settings are estimated more accurately.

The python code base containing the effect parameter estimation as well as our implementation of the effect recognition is available at:

https://github.com/henrikjuergens/guitar-fx-extraction/

9. REFERENCES

[1] Brecht De Man, Joshua Reiss, and Ryan Stables, "Ten years of automatic mixing," in *3rd Workshop on Intelligent Music Production*, 09 2017.

[2] Romain Hennequin, Anis Khlif, Felix Voituret, and Manuel Moussallam, "Spleeter: A fast and state-of-the art music source separation tool with pre-trained models," Late-Breaking/Demo ISMIR 2019, November 2019, Deezer Research.

[3] Fabian-Robert Stöter, Stefan Uhlich, Antoine Liutkus, and Yuki Mitsufuji, "Open-unmix - a reference implementation for music source separation," *Journal of Open Source Software*, vol. 4, pp. 1667, 09 2019.

[4] Christian Dittmar, Estefanía Cano, Jakob Abeßer, and Sascha Grollmisch, "Music Information Retrieval Meets Music Education," in *Multimodal Music Processing*, Meinard Müller, Masataka Goto, and Markus Schedl, Eds., vol. 3 of *Dagstuhl Follow-Ups*, pp. 95–120. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2012.

[5] Udo Zölzer, Ed., *DAFX: Digital Audio Effects*, Wiley, second edition, 2011.

[6] Kerstin Dosenbach, Wolfgang Fohl, and Andreas Meisel, "Identification of Individual Guitar Sounds by Support Vector Machines," Tech. Rep., 2008.

[7] Perfecto Herrera, Geoffroy Peeters, and Shlomo Dubnov, "Automatic classification of musical instrument sounds," *Journal of New Music Research*, vol. 32, 08 2010.

[8] Michael Stein, Jakob Abeßer, Christian Dittmar, and Gerald Schuller, "Automatic Detection of Audio Effects in Guitar and Bass Recordings," in *Audio En-*

gineering Society Convention 128, 2010, Data retrieved from https://www.idmt.fraunhofer.de/en/business_units/m2d/smt/guitar.html.

[9] Michael Stein, "Automatic detection of multiple, cascaded audio effects in guitar recordings," *13th International Conference on Digital Audio Effects, DAFx 2010 Proceedings*, 01 2010.

[10] Maximilian Schmitt and Björn Schuller, "Recognising Guitar Effects - Which Acoustic Features Really Matter?," in *INFORMATIK 2017*, Maximilian Eibl and Martin Gaedke, Eds. 2017, pp. 177–190, Gesellschaft für Informatik, Bonn.

[11] Matthew John Yee-King, Leon Fedden, and Mark D'Inverno, "Automatic Programming of VST Sound Synthesizers Using Deep Networks and Other Techniques," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 2, pp. 150–159, 2018.

[12] Di Sheng and György Fazekas, "Automatic control of the dynamic range compressor using a regression model and a reference sound," *20th International Conference on Digital Audio Effects, DAFx 2017 Proceedings*, 09 2017.

[13] Di Sheng and Gyorgy Fazekas, "A feature learning siamese model for intelligent control of the dynamic range compressor," *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2019.

[14] Brian McFee, Vincent Lostanlen, Matt McVicar, Alexandros Metsai, Stefan Balke, Carl Thomé, Colin Raffel, Dana Lee, Frank Zalkow, Kyungyun Lee, Oriol Nieto, Jack Mason, Dan Ellis, Ryuichi Yamamoto, Eric Battenberg, Victor Morozov, Rachel Bittner, Keunwoo Choi, Josh Moore, Ziyao Wei, Scott Seyfarth, Nullmightybofo, Pius Friesch, Fabian-Robert Stöter, Darío Hereñú, Thassilo, Taewoon Kim, Matt Vollrath, Adam Weiss, and Adam Weiss, "librosa/librosa: 0.7.1," 2019.

[15] Paul Boersma and David Weenink, "PRAAT, a system for doing phonetics by computer," *Glot international*, vol. 5, pp. 341–345, 2001.

[16] F Pedregosa, G Varoquaux, A Gramfort, V Michel, B Thirion, O Grisel, M Blondel, P Prettenhofer, R Weiss, V Dubourg, J Vanderplas, A Passos, D Cournapeau, M Brucher, M Perrot, and E Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[17] Léon Bottou and Chih-Jen Lin, "Support vector machine solvers," *Large Scale Kernel Machines*, 01 2007.

[18] Pierre Geurts, Damien Ernst, and Louis Wehenkel, "Extremely randomized trees," *Mach. Learn.*, vol. 63, no. 1, pp. 3–42, Apr. 2006.

[19] Donald Specht, "A general regression neural network," *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, vol. 2, pp. 568–576, 1991.

[20] François Chollet and Others, "Keras," https://keras.io, 2015.

[21] Eric Humphrey, Juan Bello, and Yann Lecun, "Moving beyond feature design: Deep architectures and automatic feature learning in music informatics," *Proceedings of the 13th International Society for Music Information Retrieval Conference, ISMIR 2012*, 01 2012.