Classification of Guitar Effects and Extraction of their Parameter Settings from Instrument Mixes Using Convolutional Neural Networks

Reemt Hinrichs[1], Kevin Gerkens[1], and Jörn Ostermann[1]

[1]Institut für Informationsverarbeitung, Leibniz University Hannover, Germany
hinrichs@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. Previous research focused on the classification of guitar effects and extraction of their parameter settings from solo guitar audio recordings. However, more realistic is the classification and extraction from instrument mixes. This work investigates the use of convolution neural networks (CNNs) for classification and extraction of guitar effects from audio samples containing guitar, bass, keyboard and drums. The CNN was compared to baseline methods previously proposed like support vector machines and shallow neural networks together with predesigned features. The CNN outperformed all baselines, achieving a classification accuracy of up to 97.4 % and a mean absolute parameter extraction error of below 0.016 for the distortion, below 0.052 for the tremolo and below 0.038 for the slapback delay effect achieving or surpassing the presumed human expert error of 0.05.

Keywords: convolutional neural networks, guitar effects, parameter extraction, music information retrieval

1 Introduction

Audio effects are a wide-spread tool used in the production and creation of music. They find application in all kinds of music and are applied to virtually all kinds of instruments such as vocals, guitar, keyboard and so on. In the domain of guitar-centered music, a prominent and well known effect is the overdrive effect, closely related to the distortion effect. A multitude of other effects exist such as phaser, delay, ring-modulator and many more. Several professional guitarists use guitar effects to create an unique, distinctive sound strongly associated with the artist. For the production of music and the creative process of writing music, automatic creation of guitar effects that yield a desired sound can be of interest. For this purpose, extraction algorithms are required to map audio recordings to effect classes and associated parameter settings. Early work in this domain

focused solely on the classification of guitar effects. Stein et al. pioneered this area of research with their fundamental work [7, 8] using a large set of audio features and a support vector machine to classify eleven guitar effects achieving a mean accuracy of 97.7 % for solo guitar recordings.

Further work regarding the classification of guitar effects was done by Eichas et al. [2], and Schmitt et al. [6], the latter investigating the importance of audio features and comparing the so called bag-of-audio-words approach to the use of functionals. They found both approaches to achieve similar high performance. Research regarding the extraction of guitar effect parameter settings is scarce. So far only two previous works exist: Jürgens et al. [4] pioneered this task using shallow neural networks combined with specifically design selected features for each guitar effect achieving or surpassing the (presumed) performance of a human expert. Comunitá et al. [1] used convolutional neural networks (CNNs) to extract the parameter settings of different implementations of distortion, overdrive and fuzz guitar effect plug-ins from monophonic and polyphonic guitar recordings. However, none of these research papers [1, 4] considered guitar effect parameter setting extraction from instrument mixes, i.e. audio recordings or signals, in which several instruments play at once. This is the focus of this manuscript: the classification of guitar effects from instrument mixes with emphasis on the extraction of the respective effect parameter settings. For this purpose, a custom dataset was created consisting of instrument mixes of guitar, bass, keyboard and drums. As baseline approach for classification we used the method of Stein et al. [8] and for extraction we used the method of Jürgens et al. [4] and compared it to the performance of a CNN at different volume levels of the instrument mix. Four different time-frequency representations were assessed with respect to the achieved CNN performance. To shorten the phrasing a little, guitar effect parameter setting extraction will be called effect parameter extraction or just parameter extraction.

Section 2 describes the datasets used for classification and parameter extraction, the time-frequency representations used and the training and evaluation of the CNNs. Section 3 reports the classification, extraction and robustness results. Section 4 discusses and interpretes these results and the manuscript is concluded in Section 5.

2 Method and Materials

Two datasets were created specifically for the investigations of this work, one for guitar effect classification, abbreviated GEC-GIM (for Guitar Effect Classification - Guitar Instrument Mix), and one for guitar effect parameter extraction, abbreviated GEPE-GIM (for Guitar Effect Parameter Extraction - Guitar Instrument Mix). The motivation to create two separate datasets was the large number of samples that had to be created if the same amount of parameter settings used for GEPE-GIM had been used for GEC-GIM. The very same effect plugins of Stein et al. [8] and Jürgens et al. [4] were used in this work. Effect descriptions and their parameters can be found in [8, 4, 10]. Additionally, to fur-

Fig. 1: Tablature showing the drum pattern used in all samples. All other instruments played a single note for the duration of two seconds, i.e. one bar.

ther test the capability of the CNN, the IDMT-SMT dataset from Stein et al. [4] was used for classification.

2.1 Dataset for Guitar Effect Parameter Extraction

The dataset used in this work to investigate guitar effect parameter extraction consisted of instrument mixes of guitar, keyboard, bass and drums. It was specifically created for the investigations described in this manuscript. All instruments were virtual instruments and created using the following plugins: Sonivox Bright Electric Guitar and Ample Guitar LP (guitar), Bitsonic Keyzone Classic (keyboard), Ample Bass P Lite (bass) and Manda Audio MT POWER DrumKit 2 (drums). All plugins were sample-based and thus could be expected to create realistic waveforms/sounds. While guitar, keyboard and bass played a single note in each sample, starting at the exact same time lasting for two seconds, the drums played the pattern depicted in Figure 1. The guitar and keyboard played the notes E2 and E3, bass played the notes E1 and E2. The guitar note was independently varied from keyboard and bass, while keyboard and bass notes were always moved together by an octave to reduce the amount of data. Keyboard, bass and drums were mixed at different volume levels with the guitar according to

$$premix(t) = b(t) + k(t) + d(t) \tag{1}$$

and

$$mix(t) = g(t) + \alpha \cdot premix(t), \tag{2}$$

where $b(t)$, $k(t)$, $d(t)$ and $g(t)$ are the bass, keyboard, drum and guitar signals, respectively. The parameter α controlled the mixing volume and was set such that the ratio $\frac{\max_t \alpha |premix(t)|}{\max_t |g(t)|}$ corresponded to the desired mixing volume. The volume mixes used were -36 dB, -24 dB, -12 dB, -6 dB, -3 dB, 0 dB and +3 dB with respect to the peak amplitude of the guitar waveform, i.e. at +3 dB, the peak amplitude of the premix after scaling was 3 dB larger than the peak amplitude of the guitar. Subjectively, the mix was considered realistic in the sense that keyboard and bass were clearly and loudly audible, not overshadowing each other, and the guitar clearly moved towards the background with increasing mixing volume. An example waveform of an audio sample with tremolo on the guitar showing the lowest and highest mixing volume after normalization is depicted in Figure 2. In total, three guitar effects - distortion, tremolo and slapback delay -

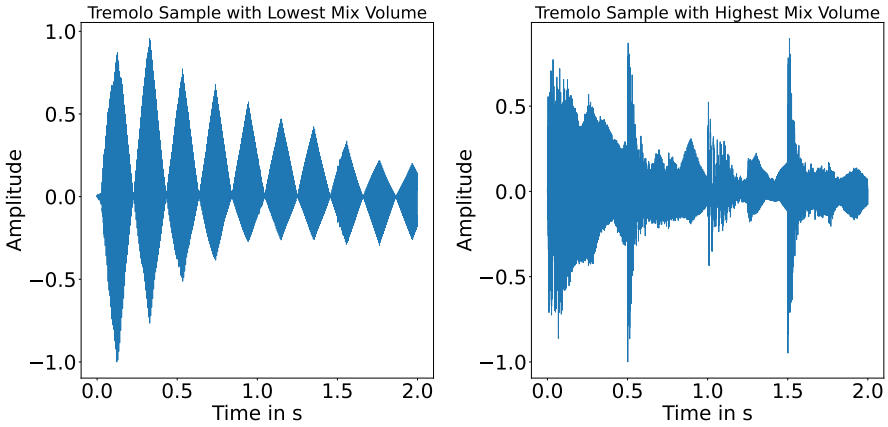


Fig. 2: Audio sample with tremolo on the guitar at (left) lowest (-36 dB) and (right) highest (+3 dB) mixing volume after peak normalization. The waveforms were peak normalized before applying the respective time-frequency representation. The two large peaks in the right figure are due to the snare hits. The depth parameter was set to 1.0 and the frequency parameter set to 0.1.

were considered and each guitar effect was applied at every mixing level. These guitar effects were chosen to allow a direct comparison to Jürgens et al. [4]. Each of these guitar effects had two parameters. These parameters were varied in steps of 0.05, starting at 0.05 and ending at 1, creating a grid of parameter values. In total 67,200 audio samples, 22,400 per effect, were created. The audio waveforms were sampled at 44.1 kHz and had a duration of two seconds each corresponding to four quarter notes or one bar at 120 beats per minute and quarter time.

2.2 Dataset for Guitar Effect Classification

The dataset for guitar effect classification from instrument mixes used exactly one fixed volume mix of 0 dB, where the mix was created as described in Section 3.1. However, in contrast, a total of eleven guitar effects were used, each using random parameter settings. Furthermore, the dataset consisted of all possible instrument mixes of the guitar and the other instruments, e.g. guitar and keyboard, guitar and bass, guitar and bass and keyboard and so on. Also solo guitar and guitar together with individual drum parts were included, e.g. guitar and snare, guitar and crash cymbal etc. This way, twelve instrument combinations were included yielding a total of 15,840 audio samples. For each combination of guitar effect, guitar plugin and instrument mix, 60 samples using random guitar effect parameter settings were generated.

2.3 Time-Frequency Representations

Four different time-frequency representations were investigated as input of the CNNs. These were the (magnitude) spectrogram, the chromagram, mel-frequency cepstral coefficients (MFCCs) and gammatone frequency cepstral coefficients (GFCCs). The chromagram is a mapping of an audio waveform to the twelve semitones of the western music that yields the energy in the respective semitones. See [5] for a description of the chromagram and the spectrogram and [3] for a description of the MFCCs and GFCCs. For the MFCCs and GFCCs, 40 coefficients were used. The spectrogram, chromagramm and the MFCCs were computed using the python framework librosa and the GFCCs using the python framework Spafe. In all cases default settings were used. For the computation of the GFCCs the sampling rate had to be reduced to 16 kHz. An example of the four time-frequency representations applied to the same audio sample is shown in Figure 3. The audio sample was an audio mix at maximum mixing volume with distortion applied to the guitar. The hi-hat is clearly visible in the spectrogram and the snare hits are apparent in the Chromagram. The input dimensions of the images obtained by applying the time-frequency representations to the audio samples were 256 x 173 (spectrogram), 40 x 173 (MFCCs), 12 x 173 (chromagram) and 40 x 193 (GFCCs). Before being fed to the CNN, the data obtained by applying the respective time-frequency representations to the audio samples was normalized to have zero mean and unit variance using sklearn's standardscaler class.

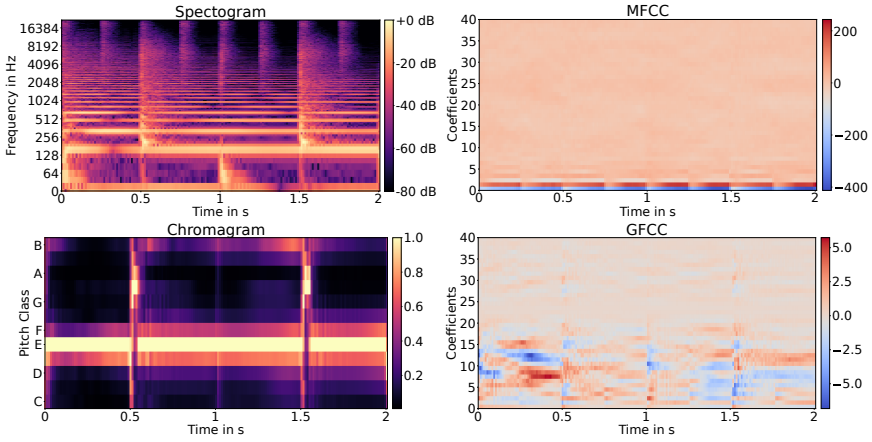


Fig. 3: Example of all four investigated time-frequency representations of one sample of the GEPE-GIT dataset with distortion effect on the guitar and gain set to 0.7 and tone set to 0.85. The snare hits are clearly visible in the chromagram and the hi-hat is apparent in the spectrogram. Mel-Frequency cepstral coefficients (MFCCs) and gammatone frequency cepstral coefficents (GFCCs) are less obvious. Except for the spectrogram, all plots use linear scale.

Table 1: Structure of the convolutional neural networks (CNNs) used for effect classification. For parameter extraction, only the number of filters, the dropout probability and number of outputs changed.

Layer	**Kernel**	**Filter**	**Activation**	**Dropout**
Convolutional	3×3	32	ReLU	-
Batch Norm.	-	-	-	-
Max Pooling	2×2	-	-	-
Convolutional	3×3	64	ReLU	0.3
Batch Norm.	-	-	-	-
Max Pooling	2×2	-	-	-
Flatten	-	-	-	-
Dense	-	64	ReLU	0.3
Batch Norm.	-	-	-	-
Dense	-	64	ReLU	0.3
Batch Norm.	-	-	-	-
Dense (Output)	-	11	Softmax	-

2.4 Convolutional Neural Networks

Two slightly different CNNs were used, one for classification of the guitar effects and one for the parameter extraction. The CNN structure used for classification is given in Table 1. For guitar effect parameter extraction, the same structure was used except with six and twelve filters in the convolutional layers as well as a dropout probability of 0.2. The total number of weights of the CNN used for classification ranged from 192,587 (Chromagram) to 10,436,683 (Spectrogram) with 1,368,139 for MFCCs and 1,597,515 for GFCCs. The total number of weights of the CNN used for parameter extraction ranged from 37,146 (Chromagram) to 1,957,914 (Spectrogram) with 257,562 for MFCCs and 300,570 for GFCCs. In the case of effect classification, the output dimension was eleven, matching the number of effects. In the case of effect parameter extraction, the output dimension was two, matching the number of parameters per effect.

2.5 Training and Evaluation

The CNNs were trained for 70 epochs with a learning rate of 0.001 using the adam solver. It was confirmed by selected visual inspection that the training

had converged after 70 epochs. A 80%/20% training/validation split of the entire dataset was used where the validation set was only used for evaluating the CNNs performance and did not influence the training process in anyway. Five repetitions, i.e. five random initializations with subsequent training of the CNNs were performed, and the results reported are an average of the results of these individual CNNs. Each training used a new training/validation split resulting in 5-fold cross validation. The batch size was set to 64 for classification and 128 for parameter extraction. Training time for the CNN was about 15 hours per repetition compared to about half an hour for the SVM.

2.6 Baseline

For effect classification, the support vector machine (SVM) classifer as described in [8] and as implemented by [4] was used, albeit the onset detection was removed as the onset was always the same due to the usage of virtual instruments in this work. Using the features and functionals proposed in [8], a total of 649 functionals were used. For effect parameter extraction, the method proposed by Jürgens et al. [4] served as baseline.

2.7 Robustness Analysis

As artificial neural networks are prone to overfitting or to fit to unexpected patterns in the data, in the most extreme case relying on isolated pixels [9], the robustness or sensitivity of the CNN for parameter extraction was analyzed. For this purpose, zero mean white gaussian noise was added to the time-frequency representations with a standard deviation σ_s set according to

$$\sigma_s = \alpha \cdot \max\{|C_s(t,f)|\}, \tag{3}$$

with $\alpha \in \{0, 0.001, 0.01, 0.05\}$ and the respective time-frequency representations $C_s(t,f)$ and its impact on the parameter extraction error was observed. The maximum was taken across the entirety of the dataset but separately for each frequency bin. An α value of zero corresponded to the original, noise-free samples and was included as reference. The index s denotes the respective time-frequency representation. Additionally, the CNN was tested with novel tones of keyboard and bass which now were moved, together, in semitone steps from $E2$ (keyboard) and $E1$ (bass) to the next octave and the impact on the parameter extraction error was assessed. An example of the impact of the noise on the spectrogram for $\alpha = 0.01$ is depicted in Figure 4. Through spectrogram inversion with the ground truth phase and the noise corrupted amplitude spectrum the corresponding audio data was subjectively judged for the spectrogram. It was found that even at the smallest noise level investigated the noise was clearly audible.

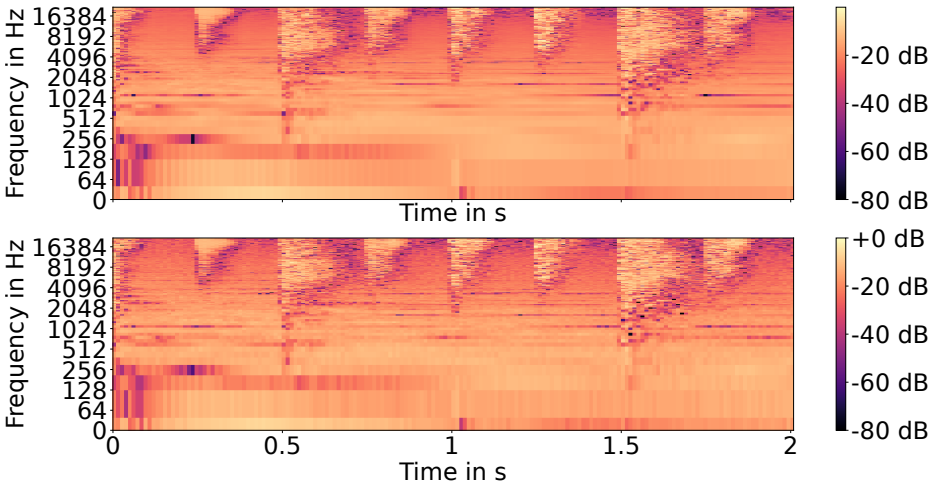


Fig. 4: Spectrogram of an audio sample at 0 dB mixing volume using distortion with gain set to 0.15 and tone set to 0.8 after normalization of the dataset as explained in Sec. 2.3 without (top) additional noise, i.e. $\alpha = 0$, and with (bottom) additional noise with $\alpha = 0.001$ and α as explained in Sec. 2.7. The noise is noticeable mostly at low magnitudes.

3 Results

The results are presented in three steps: First, the effect classification is presented. Secondly, the effect parameter extraction is presented. Finally, results regarding the robustness of the CNN are presented.

3.1 Effect Classification

The confusion matrices of the SVM and CNN classifiers on the IDMT-SMT and GEC-GIM dataset are depicted in Fig. 5. For the CNN, the depicted confusion matrices were achieved using GFCCs (IDMT-SMT) and the spectrogram (GEC-GIM). The accuracies for the other time-frequency representations were similar except for the GFCCs on the GEC-GIM dataset where the CNN failed to converge. While no obvious reason was found, inspection of the loss curves suggested an insufficient learning rate. However, because the GFCCs otherwise performed very well, this issue was not investigated further. Table 2 summarizes the accuracies of the CNN and SVM classifier for both datasets with respect to the time-frequency representation. On the GEC-GIM, both classifiers tended to confuse slapback and feedback delay which greatly impacted the overall accuracies. However, it was later realized that the feedback delay plugin at least up to a setting of 0.3 of the feedback parameter was very hard to distinguish from the slapback delay. Due to the random settings used, about 30 % of all feedback delay samples used a value of 0.3 or less making feedback and slapback delay

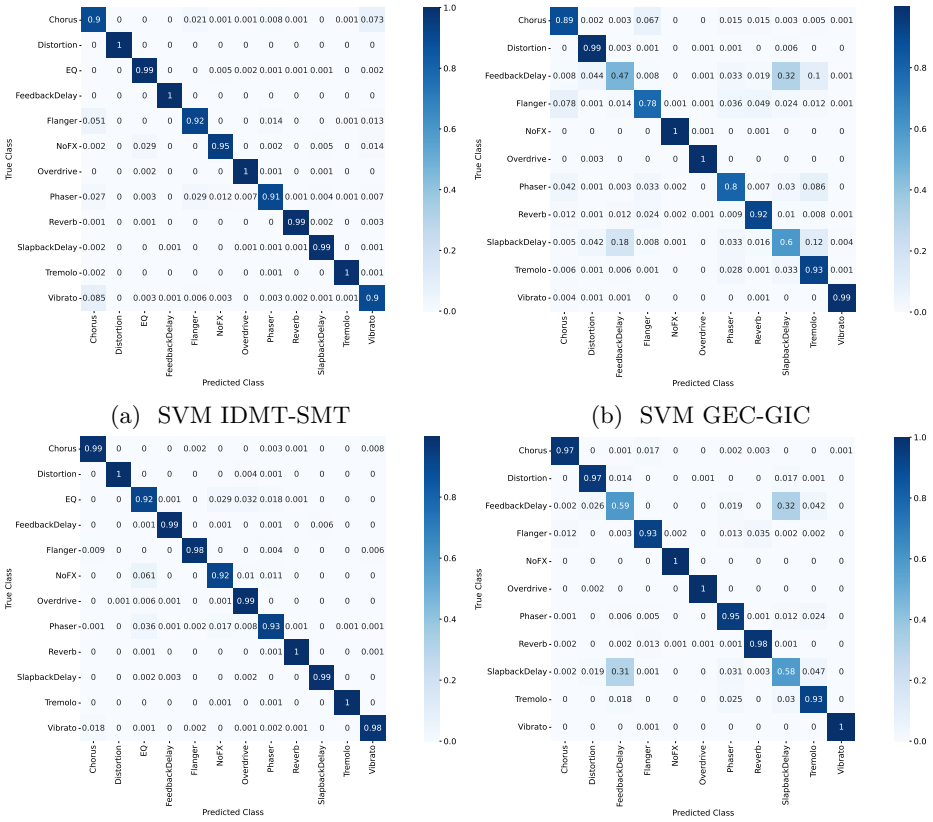
Table 2: 95 % confidence intervals for the classification accuracy of the convolutional neural network (CNN) using the listed time-frequency representations as well as the accuracy of the baseline support vector machine (SVM) using the method by Stein et al. [8]. In all cases the CNN outperformed the SVM where the highest accuracy is highlighted using bold font. As subsequent parameter extraction is not necessarily compromised by a confusion of slapback delay (SD) and feedback delay (FD), the respective accuracies, when SD and FD are treated as the same effect, are given as well.

Method	**GEC-GIM**	**GEC-GIM (SD = FD)**	**IDMT-SMT**
SVM	85.0 % ± 0.44 %	89.7 % ± 0.44	96.1 % ± 0.3 %
CNN + Spectrogram	**90.0** % ± 0.57 %	**95.7** % ± 0.57 %	**97.4** % ± 0.7 %
CNN + MFCCs	87.7 % ± 0.52 %	93.0 % ± 0.52 %	96.5 % ± 0.13 %
CNN + GFCCs	24.7 % ± 27.3 %	28.6 % ± 27.3 %	**97.4** % ± 0.3 %
CNN + Chromagram	87.0 % ± 0.64 %	92.8 % ± 0.64 %	86.2 % ± 0.2 %

virtually indistinguishable for at least these samples as confirmed by selected manual inspection. This issue did not arise on the IDMT-SMT, where rather distinct settings for slapback and feedback delay were used. On both datasets, the CNN outperformed the SVM classifier with an accuracy of up to 90.02 % for the CNN on the GEC-GIM and 85.01 % for the SVM. On the IDMT-SMT, the CNN achieved up to 97.38 % accuracy in contrast to 96.16 % accuracy for the SVM classifier. As the slapback delay is identical to the feedback delay when the feedback parameter is set to zero, confusing these two effects is not necessarily a problem for subsequent parameter extraction. Due to this, the classification accuracy when these two are considered as the same effect was specified as well increasing the accuracies by about 4-5 % in all cases.

3.2 Effect Parameter Extraction

Boxplots of the absolute error of the parameter extraction across all volumes for the CNN and the four different input representations as described in Section 2.3 as well as the method by Jürgens et al. [4] are shown for the distortion effect and its gain parameter in Figure 6a, for the tremolo effect and its frequency parameter in Fig. 6b and the slapback delay effect and its time parameter in Fig. 6c. The horizontal lines inside of the boxes denotes the median. Additionally, the presumed human expert error of 0.05 was included as a reference, see also Jürgens et al. [4]. From our own experience of creating guitar sounds, we estimated that a guitar player can be sufficiently accurate by only setting effect parameters in steps of 0.1, which would result in a minimum absolute error of 0.05. The mean absolute parameter extraction error across all volumes is summarized in Table 3. No single time-frequency representation was optimal irrespective of the considered guitar effect. The CNN using MFCCs achieved the minimum error for the distortion effect of below 0.017 for either parameter, the GFCCs worked best for the slapback delay yielding a mean error below 0.04 for either parameter.



(a) SVM IDMT-SMT

(b) SVM GEC-GIC

(c) CNN IDMT-SMT

(d) CNN GEC-GIC

Fig. 5: Confusion matrices of the support vector machine classifier (SVM) on the (a) IDMT-SMT and (b) GEC-GIC dataset as well as the convolutional neural network (CNN) classifier on the (c) IDMT-SMT and (d) GEC-GIC dataset. On the GEC-GIC both classifiers tended to confuse slapback and feedback delay which was due to unfortunate random settings of the feedback parameter which always was set to rather low values making these effects very difficult to distinguish which was confirmed through selected manual inspection. On the IDMT-SMT, which used very distinct parameter settings no confusion occured for either classifier.

The outliers occured mostly for effects and their parameter settings that were very difficult to hear, e.g. any tone setting of the distortion effect when the gain was very small. Then the tone parameter has almost no impact on the sound. For the slapback delay and its time parameter the maximum of the absolute error across the five training repetitions is depicted across the true parameter settings in Fig. 7. There, the maximum errors tended to occur for the smallest value of the mix parameter where the delay is almost inaudible.

The mean absolute extraction error across mixing volume for the slapback delay is shown in Fig. 6d. The other effects qualitatively showed similiar dependency on the mixing volume. Usually a two or three fold increase from the lowest to the highest mixing volume was observed albeit even at the highest mixing volume of +3 dB the mean performance was, on average, at or below human expert level. The method by Jürgens et al. was found to be less affected by the mixing volume showing only a minor increase of the mean absolute error with increasing mixing volume but still performed considerably worse than the CNN even at the highest mixing volume. Sample files showcasing the effect parameter extraction can be found under https://bit.ly/3nIEv8U.

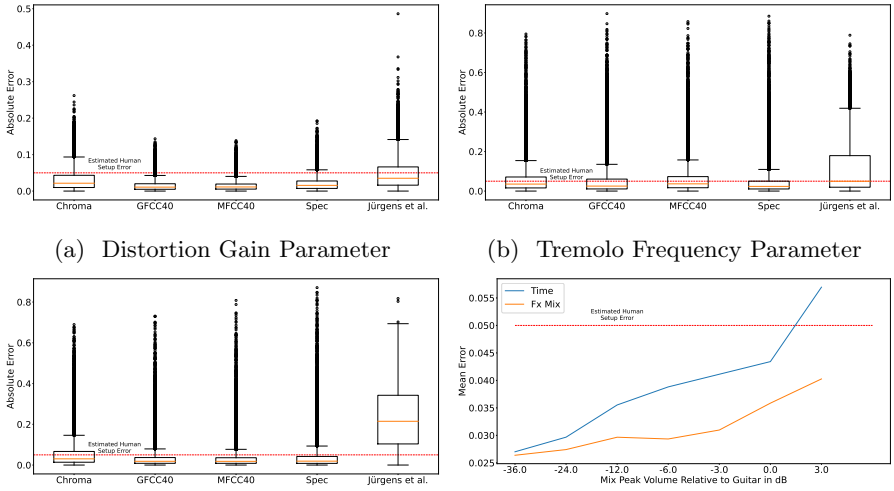
3.3 Robustness to Noise and Pitch Shifts

The mean absolute error across noise levels of the effect parameter extraction for all time-frequency representations is shown in Figure 8a for the time parameter of the slapback delay. The mixing volume in this investigation was always 0 dB. For the MFCCs, GFCCs and the Chromagram, the mean absolute error was found to be relatively robust to additive gaussian noise with an approximate increase of the mean absolute error by around 0.01-0.03 depending on the guitar effect and time-frequency representation considered. The spectrogram proved to be rather sensitive showing a considerable increase of about 0.05-0.15 depending on the guitar effect. Qualitively, the noise impact was similiar for the other parameters and effects as well.

The mean absolute error across pitch is depicted in Fig. 8b at a mixing volume of -12 dB. None of the pitches between the outer two notes was part of the training data. GFCCs allowed to achieve a mean absolute error of about

Table 3: 95 % confidence interval of the mean absolute error across the five repetitions of the parameter extraction across all volumes of the CNN and the respective time-frequency representations as well as the method by Jürgens et al. [4]. The lowest errors are highlighted using bold font.

	Distortion		Tremolo		Slapback Delay	
	Gain	Tone	Depth	Freq.	Time	Mix
Chromagram	0.032 ± 0.0015	0.051 ± 0.0048	0.063 ± 0.0017	0.061 ± 0.0038	0.062 ± 0.0025	0.054 ± 0.003
GFCC	0.016 ± 0.0025	0.017 ± 0.0015	**0.034** ± 0.002	0.063 ± 0.0033	**0.038** ± 0.0007	**0.027** ± 0.0014
MFCC	**0.014** ± 0.0009	**0.016** ± 0.0009	0.049 ± 0.0045	0.065 ± 0.0015	0.039 ± 0.001	0.031 ± 0.0022
Spectrogram	0.021 ± 0.0063	0.023 ± 0.0018	0.039 ± 0.0025	**0.052** ± 0.0059	0.049 ± 0.0009	0.037 ± 0.0047
Jürgens et al.	0.048 ± 0.0082	0.071 ± 0.024	0.129 ± 0.034	0.118 ± 0.0208	0.228 ± 0.076	0.207 ± 0.0061



(a) Distortion Gain Parameter

(b) Tremolo Frequency Parameter

(c) Delay Time Parameter Parameter

(d) Error across Volume for the Slapback Delay

Fig. 6: Boxplots of the absolute parameter extraction error for the (a) distortion gain, (b) tremolo frequency and (c) slapback delay time parameter for all investigated methods across all volumes. The outliers occured most frequently for parameter settings where the respective parameter had little to no effect due to interference of the second effect parameter. (d) shows the mean parameter extraction error across volume for the slapback delay.

0.1 or less for all pitches at -12 dB, performance was considerably worse, often above 0.15 or 0.2, when the spectrogram or the chromagram was used. At a mixing volume of -3 dB, the general performance decreased even further and, at +3 dB, parameter extraction failed almost entirely for any pitch between the two E notes, except for the gain parameter of the distortion effect, where errors below 0.1 were still achieved for the GFCCs and MFCCs. For all other effects and parameters, the mean error rose up to about 0.4 or higher at +3 dB mixing volume.

4 Discussion

Generally, the CNN was found to be superior to the investigated baselines in both effect classification and effect parameter extraction. All time-frequency representations allowed to achieve an accuracy around or better than human expert level in parameter extraction with the chromagram performing the worst out of the four. MFCCs and GFCCs were found to perform approximately the same, neither having a clear edge over the other.

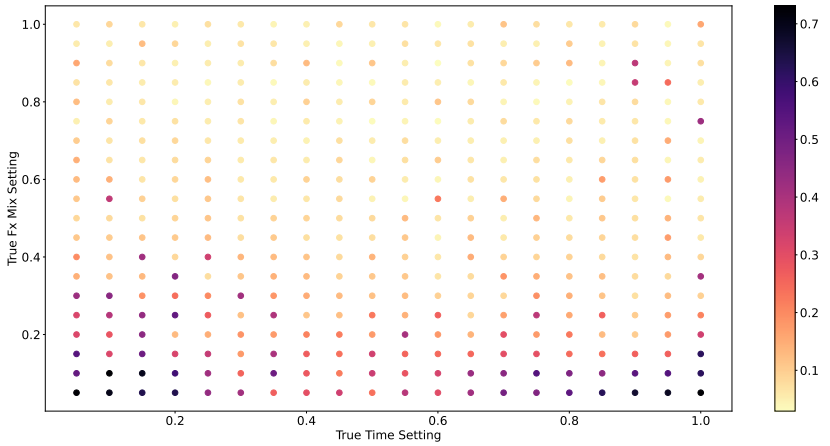
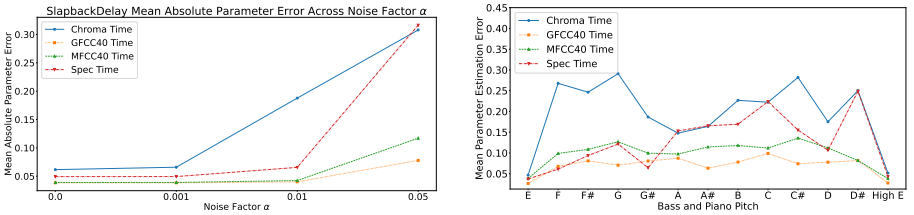


Fig. 7: Maximum extraction error for the time parameter across all five repetitions of the absolute parameter extraction error across true mix and time settings of the slapback delay. Most of the large errors occured at the smallest mix setting where the effect is very difficult (or potentially impossible) to hear and furthermore occurs also where the time parameter is very small or very large which adds to the extraction difficulty.

4.1 Guitar Effect Classification

The CNN outperformed the SVM classifier on both datasets except when the chromagram was used. The chromagram contains the most coarse information about the audio out of all investigated time-frequency representations as it maps different octaves to the same pitch value therefore it performing the worst was not surprising. The CNN achieved in its best configuration an accuracy of 97.4 % on the IDMT-SMT which is very close to the 97.7 % reported by Stein et al. [8] in their SVM implementation.

As the CNN achieved good results for both datasets, one of them using virtual instruments, albeit sample based, with all kinds of instrument mixes and the other real recordings and a large variety of solo guitar pitches, it can be assumed that the CNN is suitable for a wide range of audio data. The confusion of slapback and feedback delay on the GEC-GIT is not surprising as mostly very small parameter settings for the feedback parameter of the feedback delay were randomly selected making it very hard or impossible to distinguish it from the slapback delay. The CNN and the SVM both failing to distinguish the two effects on the GEC-GIT and not failing on the IDMT-SMT is a strong indicator that the issue lies in the data and not the classifiers. Another approach to the effect classification and extraction problem from instrument mixes could be the application of source separation algorithms followed by some classifiers. However, initital tests for our research [4] did not yield promising results and thus this approach was not pursuit further.



(a) Error across Noise Levels (b) Error across Pitch

Fig. 8: Mean absolute parameter extraction error of the time parameter of the slapback delay across (a) noise levels and (b) pitch of keyboard and guitar and all four time-frequency representations. The impact of the noise was investigated at a mixing volume of 0 dB, the error across pitch is depicted for a mixing volume of -12 dB.

4.2 Guitar Effect Parameter Extraction

The impact of the mixing volume on the parameter extraction error qualitatively was as expected, with the error increasing generally with increasing mixing volume. The spectrogram was found to perform the best, albeit MFCCs and the chromagramm achieved similar accuracies. Only the chromagram showed a clearly inferior performance compared to the other time-frequency representations. For all effects and volumes the CNN outperformed the approach of Jürgens et al. [4] albeit the CNN showed a greater sensitivity towards the mixing volume. While the boxplots revealed considerable outliers, usually about 75 % or more of the extraction errors were below the error of a human expert. Outliers, as suggested by Fig. 7, usually occured at the highest mixing volume and at settings that were very difficult to distinguish, because some parameters have an impact on each other and render the other virtually useless at certain settings. Nonetheless, some isolated large errors occur at apparently random parameter settings. These large errors could indicate an insufficient amount of data or suboptimal data creation.

4.3 Robustness

As the CNN showed only a minor decrease in accuracy when small amounts of noise ($\alpha = 0.001$) were added to the input time-frequency representations, over-fitting at least to the individual pixels seems unlikely which some deep networks are susceptible to [9]. Because at lower mixing volumes ($\leq -12\,dB$) robustness was also observed across pitch, it is very probable that the CNN indeed extracted meaningful features. Albeit at the largest noise level the noise was considerable, it was subjectively estimated, without performing a rigorous experiment, that at all noise levels a human expert would not see as large of an increase in parameter extraction error as the CNN. Here, the CNN potentially performed suboptimally and could be improved in the future. The chromagram

generally yielding the least robust CNN is not surprising because the energy for the chromagram in the training data was almost solely contained in the E note. This will have made the CNN focus mostly on this particular pitch. Once the other instruments changed their pitch and therefore the energy distribution this approach was prone to fail.

The extraction error for the gain parameter of the distortion effect was found to be very robust with respect to the pitch of keyboard and bass and the CNN achieved around and below 0.1 mean absolute error even at +3 dB mixing volume unlike all other effects and parameters. The reason was probably the distortion effect being the only nonlinear effect introducing novel frequencies into the audio signals. These samples were then the only ones in the training data where novel frequencies could convey information about the parameter settings. Therefore it appears reasonable that for the distortion effect the CNN learned a more diverse look at the time-frequency representations which in return allowed to be more robust when the pitch of the other instruments changed. Although the number of weights of the CNN, especially when the spectrogram was used as input, was rather large in comparison to the amount of training data, the results of the robustness analysis and the use of dropout layers make it seem unlikely that relevant overfitting occured.

4.4 Limitations

One limitation of our investigation is the simplicity of the music played by the instruments. More complex musical pieces, including polyphon guitar melodies, likely will be more challenging to extract guitar effects from. Furthermore, additional audio effects on the other instruments could interfere with the parameter extraction of the guitar effects. Also a second guitar could have a considerable impact on the classification and extraction performance.

5 Conclusion

In this work guitar effect classification and guitar effect parameter extraction with convolutional neural networks (CNNs) from instrument mixes was investigated and compared to two baselines. On two datasets, the CNN achieved classification accuracies $1-5\,\%$ above the baseline accuracy achieving up to $97.4\,\%$ accuracy. Mean parameter extraction errors of below 0.016 for the distortion, below 0.052 for the tremolo and below 0.038 for the slapback delay effect were achieved matching or surpassing the presumed human expert error of 0.05. The CNN was found to be moderately robust to noise and pitch changes of the background instrumentation suggesting that the CNN extracted meaningful features.

References

1. Comunità, M., Stowell, D., Reiss, J.D.: Guitar effects recognition and parameter estimation with convolutional neural networks. Journal of The Audio Engineering Society **69**(7/8), 594–604 (july 2021)
2. Eichas, F., Fink, M., Zölzer, U.: Feature design for the classification of audio effect units by input / output measurements. Proceedings of the 18th International Conference on Digital Audio Effects (DAFx-15) (2015)
3. Jeevan, M., Dhingra, A., Hanmandlu, M., Panigrahi, B.: Robust speaker verification using gfcc based i-vectors. In: Proceedings of the International Conference on Signal, Networks, Computing, and Systems. Lecture Notes in Electrical Engineering. Springer, New Delhi (2017). https://doi.org/10.18420/in2017_12
4. Jürgens, H., Hinrichs, R., Ostermann, J.: Recognizing guitar effects and their parameter settings. Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx2020) (2020)
5. Müller, M.: Fundamentals of Music Processing: Using Python and Jupyter Notebooks. Springer, Cham (01 2021). https://doi.org/10.1007/978-3-030-69808-9
6. Schmitt, M., Schuller, B.: Recognising guitar effects - which acoustic features really matter? In: INFORMATIK 2017. pp. 177–190. Gesellschaft für Informatik, Bonn (2017). https://doi.org/10.18420/in2017_12
7. Stein, M.: Automatic detection of multiple, cascaded audio effects in guitar recordings. 13th International Conference on Digital Audio Effects, DAFx 2010 Proceedings (2010)
8. Stein, M., Abeßer, J., Dittmar, C., Schuller, G.: Automatic detection of audio effects in guitar and bass recordings. Journal of The Audio Engineering Society (2010)
9. Su, J., Vargas, D.V., Sakurai, K.: One pixel attack for fooling deep neural networks. IEEE Transactions on Evolutionary Computation **23**(5), 828–841 (2019). https://doi.org/10.1109/TEVC.2019.2890858
10. Zölzer, U.: DAFX: Digital Audio Effects. 2nd edn. Wiley, Chichester (2011)