Machines listening to music: the role of signal representations in learning from music

Roswitha Bammer, Anna Breger, Monika Dörfler, Pavol Harar and Zdenek Smekal

March 28, 2019

1 Abstract

Recent, extremely successful methods in deep learning, such as convolutional neural networks (CNNs) have originated in machine learning for images. When applied to music signals and related music information retrieval (MIR) problems, researchers often apply standard FFT-based signal processing methods in order to create an image from the raw audio data. The impact of this basic signal processing step on the final outcome of the MIR task has not been widely studied and is not well understood. In this contribution, we study Gabor scattering and a new representation, namely mel scattering. Furthermore, we suggest an alternative enhancement of the loss function that uses transformed representations of the output data to incorporate additional available information. We show how applying various different signal analysis methods can lead to useful invariances and improve the overall performance in MIR problems by reducing the amount of necessary training data or the necessity of augmentation.

2 Introduction

Convolutional neural networks (CNNs), a class of special architectures of deep neural networks (DNNs), originated in image processing and have revolutionized the approach to many problems in machine learning. The main conceptional idea of CNNs is the introduction of locality and weight-sharing in the first layers of a DNN, in other words, to use convolutional layers. This intuitively leads to the extraction of local objects, which are searched for over the entire image using the same filter kernels, typically in a small neighbourhood of each pixel. By intermediate pooling operators, the extension of the local search increases across the layers and additionally introduces stability to local deformations. Transferring the principles of CNNs to problems in music information retrieval (MIR) has equally led to surprising successes in various applications. However, a tremendous difference may be observed in comparison to the image processing setting: in most cases, the actual signal of interest, namely the raw audio file, is not directly used as input to the network. It is first pre-processed, usually transforming the audio-file into an image allowing for some kind of time-frequency interpretation. Typical representations include the spectrogram or modifications thereof, e.g. based on mel-filtering. In this contribution, we introduce more involved representations of the input and output signals of the network and show experimentally how the usage of these representations improves performance on learning problems for audio signals.
The paper is organized as follows. In Section 3 we start with basic information about the learning setup and the data used in the numerical experiments. In Section 4 we present some basic signal representations and proceed to the definitions of Gabor scattering and mel scattering. The variational loss function applied to output data is explained in Section 5. Finally, all representations are evaluated on classification of instrumental sound in Section 6.

3 Learning from Data: Basic Principles

We assume the important case of learning from data. In this case, we are given a data set \( D \subset X \) in an input space \( X \), together with some information about the data, often called "annotation", which is given in the output space and denoted by \( T \subset Y \). Learning the relationship between \( D \) and their annotations in \( Y \) can then be understood as looking for a function \( \psi : X \rightarrow Y \), which describes with sufficient accuracy the desired mapping. Accuracy, in turn, is measured by a loss function, which is used both as a basis for training a model, that is, for learning the optimal parameters for the description of the desired mapping from input to output data and for measuring the performance of the model once a parameter vector \( \theta \) determining a particular model within a determined architecture has been learned. One assumes that data \( Z_m = D \times T = \{(x_1, y_1), \ldots, (x_m, y_m)\} \) are drawn i.i.d. from a (usually unknown) probability density \( \rho : X \times Y \rightarrow \mathbb{R} \). Further, given a hypothesis space, parametrized by \( \theta \), and a set of annotated data \( Z_m \), from which a model \( \psi_\theta \) is learned, we let the estimated targets be denoted by \( \hat{y}_i = \psi_\theta(x_i) \) and define the empirical loss function \( E_{Z_m} \) as

\[
E_{Z_m}(\psi_\theta) = \frac{1}{m} \sum_{i=1}^{m} L(y_i, \hat{y}_i).
\]

Common, important examples of loss functions include the quadratic loss \( L(y_i, \hat{y}_i) = (\hat{y}_i - y_i)^2 \), and the categorical cross-entropy loss. The latter is the concatenation of the softmax function on the output vector \( y = (\psi_\theta(x_1), \ldots, \psi_\theta(x_m)) \) and the cross-entropy loss; in other words, in the case of categorical cross-entropy (CC) loss, we have

\[
L(y_i, \hat{y}_i) = -y_i \log \frac{e^{\hat{y}_i}}{\sum_{j=1}^{m} e^{\hat{y}_j}}.
\]

3.1 Data Set used for Experiments

For the classification experiments presented in Section 6 the GoodSounds data set [16] is used. It contains monophonic recordings of single notes or scales played by different instruments. From each file, we have removed the silence with SoX v14.4.2 library [2, 14]. The output rate was set to 44.1 kHz with 16 bit precision. We have split each file into segments of the same duration (1 s = 44100 samples) and applied a Tukey window in order to smooth the onset and offset of the segment to prevent undesired artifacts after applying the short-time Fourier transform (STFT). Since the classes were not equally represented in the data set, we needed to introduce an equalization strategy. To avoid extensive equalization techniques, we have used only classes which spanned at least 10% of the whole data set, namely clarinet, flute, trumpet, violin, alto saxophone and cello. More precisely, during the process of cutting the audio samples into 1 s segments, we introduce increased overlap for instrument recordings with fewer samples, thus utilizing a variable stride. This resulted in oversampling in underrepresented classes by overlapping the segments.

4 Representing Music: Listen and Look!

Classical audio pre-processing tools such as the mel-spectrogram rely on some localized, FFT-based analysis. The idea of the resulting time-frequency representation is to separate the variability in the signal with respect to time and frequency, respectively. However, for audio signals which are relevant to human perception, such as music or speech, significant variability happens on very different time-levels: the frequency content itself can be determined within a few milliseconds.
Variations in the amplitude of certain signal components, e.g., formants or harmonics, have a much slower frequency and should be measured on the scale of up to few seconds. Longer-term musical developments, which allow, for example, to determine musical style or genre, happen on time-scales of more than several seconds. The basic idea of Gabor Scattering (GS), as introduced in [3], see Section 4.2, is to capture the relevant variability at different time-scales and separate them in various layers of the representation.

We first recall (mel-)spectrograms and turn to the definition of the scattering transforms in Section 4.2.

4.1 Spectrograms and Mel-Spectrograms

Standard time-frequency representations used in audio-processing are based on STFT. Since we are interested in obtaining several layers of time-frequency representations, we define STFT as frame-coefficients with respect to time-frequency-shifted versions of a basic window. To this end, we introduce the following operators in some Hilbert space $\mathcal{H}$.

- The translation (time shift) operator for a function $f \in \mathcal{H}$ and $t \in \mathbb{R}$ is defined as $T_x f(t) := f(t - x)$ for all $x \in \mathbb{R}$.
- The modulation (frequency shift) operator for a function $f \in \mathcal{H}$ and $t \in \mathbb{R}$ is defined as $M_\omega f(t) := e^{2\pi i t \omega} f(t)$ for all $\omega \in \mathbb{R}$.

Now the STFT $V_g f$ of a function $f \in \mathcal{H}$ with respect to a window function $g \in \mathcal{H}$ can be easily seen to be $V_g f(x, \omega) = \langle f, M_\omega T_x g \rangle$ with the corresponding spectrogram $|V_g f(x, \omega)|^2$. The set of functions $G(g, \alpha, \beta) = \{M_\beta T_\alpha g : (\alpha, \beta) \in \Lambda\}$ is a the Gabor system and is called Gabor frame [7], if there exist positive frame bounds $A, B > 0$ such that for all $f \in \mathcal{H}$

$$A\|f\|^2 \leq \sum_k \sum_j |\langle f, M_\beta T_\alpha g \rangle|^2 \leq B\|f\|^2. \quad (1)$$

Subsampling $V_g f$ on a separable lattice $\Lambda = \alpha \mathbb{Z} \times \beta \mathbb{Z}$ we obtain the frame-coefficients of $f$ w.r.t $G(g, \alpha, \beta)$. Choosing $\Lambda$ thus corresponds to picking a particular hop size in time and a finite number of frequency channels.

The mel-spectrogram (MT) $MS_g(f)$ is defined as the result of weighted averaging $|V_g f(\alpha k, \beta j)|^2$:

$$MS_g(f)(\alpha k, \nu) = \sum_j |V_g f(\alpha k, \beta j)|^2 \cdot \Upsilon_{\nu}(j),$$

where $\Upsilon_{\nu}$ are the mel-filters for $\nu = 1, ..., K$ with $K$ filters.

4.2 Gabor Scattering and Mel Scattering

We next introduce the principles of scattering transforms. Scattering transforms based on Gabor frames lead to a new feature extractor called Gabor scattering (GS), first introduced in [3]. In this contribution, we further extend the idea of Gabor-based scattering by adding a mel-filtering step in the first layer. The resulting transform is called mel scattering (MS). GS is a feature extractor for audio signals, obtained by an iterative application of Gabor transforms, a non-linearity in the form of a modulus function and pooling by sub-sampling in each layer. Since most of the energy and information of an input signal is known to be captured in the first two layers, cp. [1], we only introduce and use the output of those first layers, while in principle scattering transforms allow for arbitrarily many layers. In [3], it was shown that the output of specific layers of GS lead to invariances w.r.t. certain signal properties. Coarsely speaking, the output of the first layer is invariant w.r.t. envelope changes and mainly captures the frequency content of the signal, while the second layer is invariant w.r.t. frequency and contains information about the envelope. For
more details on GS and a mathematical description of its invariances see [3].
In the following, since we deal with discrete, finite signals \( f \), we let \( \mathcal{H} = \mathbb{C}^L \), where \( L \) is the signal length, and \( f_\ell \in \mathbb{C}^{L_\ell} \) for \( \ell = 1, 2 \). The lattice parameters of the Gabor transform, i.e. \( \Lambda_\ell = \alpha_\ell \mathbb{Z} \times \beta_\ell \mathbb{Z} \), can be chosen differently for each layer.

The first layer, which is basically a Gabor transform, corresponds to

\[
    f_1[\beta_{1,j}](k) = \langle f, M_{\beta_{1,j}}T_{\alpha_1}g_1 \rangle,
\]

and the second layer can be written as

\[
    f_2[\beta_{1,j}, \beta_{2,h}](m) = \langle f_1[\beta_{1,j}], M_{\beta_{2,h}}T_{\alpha_2}g_2 \rangle.
\]

Note that the input function of the second layer is \( f_1 \), where the next Gabor transform is applied separately to each frequency channel \( \beta_{1,j} \). To obtain the output of one layer, one needs to apply an output generating atom \( \phi_\ell \), cp. [3]:

\[
    f_\ell[\beta_{1,s}, ..., \beta_{\ell,j}] \ast \phi_\ell(k) = \langle f_{\ell-1}, M_{\beta_{\ell,j}}T_{\alpha_{\ell}}g_1 \rangle \ast \phi_\ell,
\]

for \( \ell \in \mathbb{N} \) in general and in our case \( \ell = 1, 2 \).

The output of the feature extractor is the collection of these coefficients [4] in one vector, which is used as input to a machine learning task. Based on the GS we want to introduce an additional mel-filtering step. The idea is to reduce the redundancy in spectrogram by frequency-averaging. The expression in (2) is then replaced by

\[
    f_1[v](k) = \sum_j \langle f_0, M_{\beta_{1,j}}T_{\alpha_1}\ell g_1 \rangle \cdot \Upsilon_\nu(j),
\]

where \( \Upsilon_\nu \) corresponds to the mel-filters, as introduced in Section 4.1. The other steps of the scattering procedure remain the same as for GS, i.e. performing another Gabor transform to obtain layer 2 and afterwards applying an output generating atom in order to obtain the MS coefficients. The output of GS and MS can be visually explained by Figure 1. The naming Output A displays either the output of Equation (2) in case of GS or Equation (5) in the MS case. The Output B shows the spectrogram after applying the output generating atom and Output C illustrates the output of the second layer.

5 Variational Loss Function for Sound Data

In the previous sections we introduced different input data representations, for subsequent classification via deep learning. In the following we want to investigate possible enhancement with new output/target data representations. To do so, we use a variational loss function, introduced in [4] in a general setting, which allows for the integration of additional output information via informed transformation of the target space. We now recall the non-weighted variational loss function from [4] and describe in detail, how its properties can be exploited in the setting of the current work.

Our training data is given by the MT, GS or MS computed from the annotated sounds in GoodSounds, introduced in Section 3.1. The inputs to the network are thus arrays \( \{x_i\}_{i=1}^m \subset \mathbb{R}^{120 \times 160} \) and have associated target values \( \{y_i\}_{i=1}^m \subset \{0, 1\}^6 \), corresponding to the 6 instrument classes. As exemplified in Section 3, in each optimization step for the parameters of the neural network, the network’s output \( \{\hat{y}_i\}_{i=1}^m \subset \mathbb{R}^6 \) is compared with the targets \( \{y_i\}_{i=1}^m \) via an underlying loss function \( L \). However, the training data often naturally contains additional important target information that is not used in the original representation and we aim at incorporating such information, tailored to the particular learning problem, by using the non-weighted variational loss function, which may then be written as

\[
    L_{\text{var}}(\{y_i\}, \{\hat{y}_i\}) = \sum_{j=1}^n \alpha_j L_j(\{T_j(y_i)\}, \{T_j(\hat{y}_i)\}).
\]
Here, for all \( j = 1, \ldots, n \), we let \( \alpha_j > 0 \) be an adjustable weight of \( L_j \), which is some standard loss function and \( T_j : \{0, 1\}^6 \to \mathbb{R}^t \) is a transformation which encodes the additional information on the target space. Specifically, \( T_1 \) corresponds to the identity on \( \mathbb{R}^6 \), i.e. no transformation is applied in the first component, for which \( L_1 \) is the categorical cross-entropy loss [18]. For \( j = 2, \ldots, n \), \( L_j \) is chosen as the mean squared error and in all these cases we have dimension \( t_j = 1 \) for the incorporation of additional information on the GoodSounds data set, which is described in detail in the following section.

5.1 Design of Transformations

We heuristically choose \( d = 16 \) transformations \( T_2, \ldots, T_{17} \) that use output features arising directly from the particular output class, with \( T_j : \{0, 1\}^6 \to \mathbb{R}^t \), for \( j = 2, \ldots, 17 \). Amongst others the features are chosen from the enhanced scheme of taxonomy [17] and from the table of frequencies, harmonics and under tones [19]. We choose transformations that provide natural information already contained in the underlying instrument classes. The additional terms in the loss function enable to penalize common classification errors. In this experiment, the transformations are given by the inner product of the output/target and the feature vector. E.g. we directly know to which instrument family an instrument belongs and distinguish between woodwind, brass and bowed instruments, and moreover between chordophone and aerophone instruments. Let’s assume a target vector \( y_{i(j)} = \delta_{ij} \), corresponds, respectively, to the instruments clarinet, flute, trumpet, violin, saxophone and cello, and the output of the network is \( \hat{y}_i = (a_1, a_2, a_3, a_4, a_5, a_6) \in \mathbb{R}^6 \). The feature vector \( f_1 = (1, 1, 0, 0, 1, 0) \) then captures the information "target instrument is from family woodwind". The transformation may be defined by \( T_1(y_i) = \langle y_i, f_1 \rangle \) in order to incorporate this information. Additionally, by choosing \( \alpha_j \), we can weight the amount of penalization for wrong assignments in \( (T_1(y_i) - T_1(\hat{y}_i))^2 \). Amongst others we also use minimum and maximum frequencies of the instrument as features. E.g. the feature corresponding to minimum frequency \( f_2 = (b_1, b_2, b_3, b_4, b_5, b_6) \in \mathbb{R}^6 \). Again the transformation is given by \( T_2(y_i) = \langle y_i, f_2 \rangle \). Choosing the right penalty for this feature could prohibit that instruments belonging to the same instrument family are classified wrong, e.g. a cello that would be classified as a violin.

6 Experimental Results: Application to musical data

In the numerical experiments we compare the MT with GS and MS time-frequency representations in a supervised learning problem. The overall task was a multi-class classification of musical instruments based on the audio signal. For signals from GoodSounds [16], introduced in Section 3.1, the comparison was conducted using a CNN architecture. Two loss functions were compared, namely standard categorical cross-entropy loss (CC) and variational loss (VL) as introduced in Section 5.

6.1 Computation of Signal Representations

The raw audio signals were transformed into multiple time-frequency representations, namely MT, MS and GS using the Gabor-scattering v0.0.4 library [8]. The library is our Python implementation of all mentioned signal representations with the aim to provide the community with easy access to the transformations. The library’s core algorithms are based on Scipy v1.2.1 [10] implementation of STFT and mel-filter banks from Librosa v0.6.2 library [13]. For baseline results we have used standard CC loss function as implemented in the Keras framework. Visualization of the time-frequency transformations of a random training sample can be seen in Figure 1.
Figure 1: Visualization of time-frequency transformations.
Table 1: Accuracy of classification on testing set.

| NB  | MT_{CC} | MT_{VL} | MS_{CC} | GS_{CC} |
|-----|---------|---------|---------|---------|
| 550 | 0.9340  | 0.9446  | 0.9619  | 0.9863  |
| 110 | 0.8800  | 0.8943  | 0.8927  | 0.9337  |
| 55  | 0.8391  | 0.8383  | 0.8372  | 0.9145  |
| 11  | 0.7267  | 0.7304  | 0.7532  | 0.7905  |
| 5   | 0.6193  | 0.6143  | 0.6213  | 0.6838  |

Table notation:
NB – Number of training batches. MT_{CC}, MS_{CC} and GS_{CC} – mel spectrogram, mel scattering and Gabor scattering as input representations with CC. MT_{VL} – mel spectrogram as input representation with VL.

6.2 Deep convolutional neural network

We implemented our experiment using Python 3.6. A CNN was created and trained from scratch on Nvidia GTX 1080 Ti GPU in Keras 2.2.4 framework [5] using the described training set split into batches of size 128. We used an architecture consisting of four convolutional stacks. Each of them consists of a convolutional layer, rectified-linear unit activation function and average pooling. These stacks were followed by a fully connected layer with softmax activation function. Each network had to be adjusted slightly, because the input shapes changed according to the time-frequency representation used (GS having 3 channels, MT having less frequency channels etc.). We have tried to make the results as comparable as possible, therefore the networks differ only in the number of channels of the input layer, the rest of the network is only affected by the number of frequency channels, which thanks to pooling did not cause significant difference in the number of trainable parameters. All networks have comparable number of trainable parameters within range from 7 888 to 9 860. The weights were optimized using Adam optimizer [11]. Reproducible code can be found in the repository [9].

Figure 2: CNN performance over number of batches used in training.

6.3 Training and Results

All the samples were split into training, validation and testing sets in such a way that validation and testing sets have exactly the same number of samples from each class, while this holds for
Table 2: Detailed comparison of performances.

| NB | TF | Categorical cross-entropy | Variational loss |
|----|----|----------------------------|------------------|
|    |    | train | valid | test | train | valid | test |
| 55 | GS | 0.9805 | 0.9867 | **0.9863** | 0.9791 | 0.9870 | 0.9852 |
|    | MS | 0.9449 | 0.9540 | **0.9619** | 0.9328 | 0.9440 | 0.9502 |
|    | MT | 0.9233 | 0.9304 | 0.9340 | 0.9336 | 0.9437 | **0.9446** |
| 110| GS | 0.9298 | 0.9321 | 0.9337 | 0.9287 | 0.9462 | **0.9430** |
|    | MS | 0.8776 | 0.8873 | **0.8927** | 0.8726 | 0.8860 | 0.8905 |
|    | MT | 0.8674 | 0.8785 | 0.8800 | 0.8766 | 0.8907 | **0.8943** |
| 55 | GS | 0.9168 | 0.9147 | 0.9145 | 0.9107 | 0.9160 | **0.9158** |
|    | MS | 0.8345 | 0.8294 | **0.8372** | 0.8278 | 0.8253 | 0.8306 |
|    | MT | 0.8227 | 0.8308 | **0.8391** | 0.8321 | 0.8328 | 0.8383 |
| 11 | GS | 0.7955 | 0.7889 | **0.7905** | 0.7734 | 0.7702 | 0.7732 |
|    | MS | 0.7599 | 0.7429 | **0.7532** | 0.7408 | 0.7209 | 0.7327 |
|    | MT | 0.7301 | 0.7100 | 0.7267 | 0.7251 | 0.7202 | **0.7304** |
| 5  | GS | 0.7063 | 0.6773 | **0.6838** | 0.6641 | 0.6525 | 0.6550 |
|    | MS | 0.6422 | 0.5940 | 0.6213 | 0.6375 | 0.6127 | **0.6384** |
|    | MT | 0.6219 | 0.6004 | **0.6193** | 0.6328 | 0.5983 | 0.6143 |

Table notation:
NB – Number of training batches. TF – Time-frequency representation. Train, valid and test – accuracy on training, validation and testing sets. Bold font denotes better accuracy between CC and VL.

Training set only approximately. Segments from audio files that were used in validation or testing were not used in training to prevent leaking of information. Detailed information about the used data, stride settings for each class, obtained number of segments and their split can be found in the repository [9].

In total we have trained 30 different models (3 time-frequency representations, 2 different loss functions, 5 training set sizes), with the following hyper-parameters: number of convolutional kernels in the first 3 convolutional layers is 16 each, learning rate is 0.001, \( \alpha \) of variational loss is 0.01 and \( \alpha \) of \( L_2 \) weight regularization is 0.001.

The results are shown in Table 1 and Table 2. Both tables show the accuracies of the model’s best epoch. Table 1 shows only the accuracies on testing set, while Table 2 contains information about the performance on training and validation sets as well. Best epoch was always chosen based on the performance on the validation set. Accuracy is computed as fraction of correct predictions to all predictions.

Mel spectrogram has relatively low dimensional input, therefore training of one epoch of the CNN in this setting took 30 s. In comparison, training with GS is 10 times slower, i.e. it takes 300 s to train one epoch. Nonetheless, the performance of GS for this problem was always superior to MT as seen in Table 1. In Table 2 we can see that the GS needed approximately 5 times less training data to achieve about the same accuracy as MT. In our experimental setup MS performance was usually between MT and GS, but the training was less time consuming than GS, i.e. 150 s per epoch.

6.4 Discussion of Results and Perspectives

Previous work on Gabor scattering showed that signal variability w.r.t. different time scales are separated by this transform, cf. [3], a beneficial property for learning. The most common choice of a time-frequency representation of audio signals is mel-spectrogram; hence, as a natural step, we introduced MS in this paper, a new feature extractor combining the properties of GS with mel-filter averaging. This led us to a representation with smaller dimensionality, and better
performance than MT while cutting down the training time in comparison to GS. Using the VL function in addition did not affect the speed, but the accuracy of the classification task. All new representations outperformed the MT-based input on the GoodSounds data set, cp. Table 1. Concerning the additional information provided by adding non-weighted VL, from the results in Table 2 it seems that this mainly improves the results on the simpler input transformation MT; future research will study the interplay between input and output representations in more detail.

References

[1] J. Andén and S. Mallat. Deep scattering spectrum. *IEEE Transactions on Signal Processing*, 62(16):4114–4128, 2014.

[2] C. Bagwell. https://launchpad.net/ubuntu/+source/sox/14.4.1-5 Accessed: 2018-10-31.

[3] R. Bammer, M. Dörfler, and P. Harar. Gabor frames and deep scattering networks in audio processing. *submitted*, 2019. https://arxiv.org/pdf/1706.08818.pdf.

[4] A. Breger, J. I. Orlando, P. Harar, M. Dörfler, S. Klimscha, C. Grechenig, B. S. Gerendas, U. Schmidt-Erfurth, and M. Ehler. On orthogonal projections for dimension reduction and applications in variational loss functions for learning problems. *arXiv preprint arXiv:1901.07598*, 2019.

[5] F. Chollet et al. Keras. https://keras.io 2015.

[6] D. Griffin and J. Lim. Signal estimation from modified short-time fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(2):236–243, 1984.

[7] K. Gröchenig. *Foundations of time-frequency analysis*. Applied and numerical harmonic analysis. Birkhäuser, Boston, Basel, Berlin, 2001.

[8] P. Harar. gabor-scattering. https://gitlab.com/paloha/gabor-scattering 2019.

[9] P. Harar. gs-ms-mt. https://gitlab.com/hararticles/gs-ms-mt 2019.

[10] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [Online; accessed 2019-02-01].

[11] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[12] S. Mallat. Understanding deep convolutional networks. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, 374 2065:20150203, 2016.

[13] B. McFee, M. McVicar, S. Balke, C. Thomé, V. Lostanlen, C. Raffel, ..., and A. Holovaty. librosa/librosa: 0.6.2, Aug. 2018.

[14] J. Navarrete. The SoX of Silence tutorial. https://digitalcardboard.com/blog/2009/08/25/the-sox-of-silence Accessed: 2018-10-31.

[15] A. V. Oppenheim. *Discrete-time signal processing*. Pearson Education India, 1999.

[16] O. Romani Picas, H. Parra Rodriguez, D. Dabiri, H. Tokuda, W. Hariya, K. Oishi, and X. Serra. A real-time system for measuring sound goodness in instrumental sounds. In *AES 138th Convention*, Warsaw, Poland, 07/05/2015 2015.

[17] E. M. Von Hornbostel and C. Sachs. Classification of musical instruments: Translated from the original german by anthony baines and klaus p. wachsmann. *The Galpin Society Journal*, pages 3–29, 1961.
[18] Z. Zhang and M. Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. *Advances in Neural Information Processing Systems (NIPS)*, 31, 2018.

[19] I. ZyTrax. Frequency ranges. http://www.zytrax.com/tech/audio/audio.html, 2018.