Postprint of: Koszewski D., Kostek B.: Musical Instrument Tagging Using Data Augmentation and Effective Noisy Data Processing JOURNAL OF THE AUDIO ENGINEERING SOCIETY Vol. 68, iss. 1/2 (2020), pp. 57-65
DOI: 10.17743/jaes.2019.0050

Musical instrument tagging using data augmentation and effective noisy data processing

DAMIAN KOSZEWSKI, *AES Member*, **AND BOZENA KOSTEK,** *AES Fellow*
(damkosze@multimed.org bokostek@audioacoustics.org)

Audio Acoustics Laboratory, Faculty of Electronics, Telecommunications and Informatics, Gdansk University of Technology, Gdansk, Poland

Developing signal processing methods to extract information automatically has potential in several applications, for example searching for multimedia based on its audio content, making context-aware mobile applications (e.g., tuning apps) or pre-processing for an automatic mixing system. However, the last-mentioned application needs a significant amount of research to recognize real musical instruments in recordings reliably. In this paper, we primarily focus on how to obtain data for efficiently training, validating, and testing a deep-learning model by using a data augmentation technique. These data are transformed into 2D feature spaces, i.e., mel-scale spectrograms. The Neural Network used in the experiments consists of a single-block DenseNet architecture and a multi-head softmax classifier for efficient learning with the mixup augmentation. For automatic noisy data labeling, the batch-wise loss masking, which is robust to corrupting outliers in data, was applied. To train the models, various audio sample rates and different audio representations were utilized. The method provides promising recognition scores even with real-world recordings that contain noisy data.

0 INTRODUCTION

Automatic music instrument recognition is an essential subtask in many applications regarding music information indexing and retrieval [1]-[5]. Computational auditory scene analysis (CASA) [6],[7], automatic music transcription frameworks, and content-based search systems, all find such a capability to be extremely helpful in music recognizing [8],[9]. However, musical instrument recognition has not received as much research interest as, for instance, speech and speaker recognition, even though both the amateur music lovers and professional musicians would benefit from a system automatically annotating musical content. A system of this kind may be used as a pre-stage for the automatic mixing system, if it automatically recognizes and tags musical instruments for further processing.

Early works, carried out more than two decades back, concentrated on extracting and developing features that enabled classifying musical instrument sounds, and later melodies containing a single musical instrument or a mixture of sounds [10]-[16]. A variety of features were formulated, creating the so-called dedicated parameters [3],[17]. Some of these parameters were borrowed from the speech analysis and recognition domains [13],[15],[18],[19], or investigated through the pattern-based semantic analysis of radio programs in search for salient features [20], others were more specifically related to the music area. Many of these features are inscribed into the MPEG 7 standard [21]. The majority of the recognition systems used for musical instrument sounds so far concentrate on the timbral-spectral characteristics of sounds. Discrimination is based on features such as pitch, spectral centroid, energy ratios, spectral envelopes, and mel-frequency cepstral coefficients [13],[22]. Temporal features, other than attack, duration, and tremolo, are less frequently taken into account. Classification is performed using statistical models, k-NN (k-Nearest Neighbors) classifier, HMM (Hidden Markov Model), Kohonen SOM (Self-Organizing Map), SVM (Support Vector Machine) [23],[26], as well as other machine learning algorithms, among them Neural Networks applied to deep learning [9]. A limitation of the classification task is that, in real instrument sounds, the temporal, spectral, and dynamic features are never constant. Even when the same note is being played, the spectral components change. Therefore, one has to take into consideration musical articulation and the way timbral components can vary. To develop a robust recognition system, learning algorithms should be employed as it is difficult to build a sufficiently large set of musical instrument patterns that will be stable over time. At the same time, various representations of musical instrument sounds should be presented to the classifier.

The goal of the experiments carried out is to gather music samples (i.e., real-life recordings, including noisy data resulting from improper labeling and also electronically generated music that may serve the purpose of the data augmentation) for training, validating and testing Neural Network (NN), that classifies effectively musical instruments. Also, we recall a process of constructing NN to obtain the best results, which follows a DCASE 2018 challenge paper [27],[28]. In addition, this paper shows that NN constructed for one task can be trained on different data and can still return sufficiently high classification scores.

Recent works have shown that convolutional networks (CNN) can be substantially deeper, more accurate, and efficient to train if they contain shorter connections between layers close to the input and those close to the output [29]. This is because when the information path between the input and the output layers in CNN is too long, the gradient may vanish. It is said that DenseNets solve this issue since each layer has direct access to the gradients from the loss function and the original input data [30]. Therefore, DenseNets are utilized to increase the depth of deep convolutional networks (CNN). They allow better control over the vanishing-gradient problem, strengthening feature propagation, exploiting the potential of feature reusing, and substantially reducing the number of parameters [31],[32]. Moreover, DenseNets, coupled with the mixup training technique [33], are able to learn from corrupt labels or noisy data.

The aim of this paper is to build a sufficiently large musical instrument sound representation, transform it into 2D feature space based on mel-scale spectrograms, and then classify them with a deep learning-based algorithm. This is based on Cambridge Music Technology website music tracks [34]. This music library, entitled “Mixing Secrets For The Small Studio”, was built for the purpose of practicing music mixes. It contains uncompressed WAV files (24-bit or 16-bit resolution and 44.1kHz sampling rate, the original tracks are about 3 minutes long) without additional effects or processing. Overall, this library includes 364 full multitrack projects, including audio files that belong to 20 musical genres, such as: acoustic, jazz, country, orchestral, electronica, dance, experimental, pop, singer-songwriter, alt-rock, blues, country rock, indie, funk, reggae, rock, punk, metal, hip-hop, and r&b [34].

The paper starts with a presentation of the proposed framework, showing an augmentation technique to mix two training sets linearly. Then, a full architecture of the proposed model focusing on the training efficiency against strong mixup augmentation [33] is presented. One of the pre-classification modules is the low-level signal representation. However, instead of extracting low-level parameters based on MPEG 7 standard to create feature vectors, we use two-dimensional data representation. Therefore, in the case of our study, mel-scale spectrograms are created and presented at the dense neural network input. The process of training is outlined, and the way of dealing with the noisy data is discussed. It is shown that the classification rates obtained for various musical instruments are promising; thus, this study will be further pursued in a real-life application.

1 PROPOSED FRAMEWORK - MIXUP AUGMENTATION

The main problem in machine learning is obtaining a sufficient number of examples for training, validation, and testing. If it is not possible to obtain additional real observations, a synthetic extension of the training set, the so-called dataset augmentation, is often used, resulting in new data. This process consists of generating new data by transforming existing examples.

A method to train on similar but different samples that are contained in the training set is known as data augmentation [35]. The method was formalized based on the Vicinal Risk Minimization (VRM) principle [36]. VRM requires to describe a vicinity or neighborhood around each example in the training data by employing expert knowledge to do that. This way, additional virtual examples can be drawn from the neighborhood distribution of the training examples to enlarge the training dataset [37]. However, as already mentioned, this method requires expert knowledge and the results are dataset-dependent. To address this problem, introduced was a mixup technique, which is a neural network training method that creates new samples by linear interpolation of existing samples and their labels [33]. It results in virtual training examples. The following definition was proposed by H. Zhang and collaborators [37]. Let x_i and y_i be the i-th raw input data in the training dataset and its corresponding binary label, respectively; then the mixup generates an augmented data item (vector) $\hat{x}$ which is the mixture of two original datasets as follows [37]:

$$\hat{x} = \lambda x_j + (1-\lambda) x_k \quad (1)$$

$$\hat{y} = \lambda y_j + (1-\lambda) y_k . \quad (2)$$

where λ ∈ (0,1) and (x_j, y_j) as well as (x_k, y_k) are examples drawn randomly from the training data.

This mixup technique shows improvements not only in the image classification tasks but also improves generalization on speech and tabular data [37]. It was reported that this technique provided a new state-of-the-art performance in the CIFAR-10, CIFAR100, and ImageNet-2012 image classification datasets [37]. Moreover, learning from corrupt labels or noisy data is also possible or even recommended. That is why we decided to use this mixup technique since one may easily find an analogy to the fact that an audio signal captured in real-world can be considered as a linear mixture of various source signals, in some cases adversarial ones. Based on this notion, classifying $\hat{x}$ to $\hat{y}$ could be considered a task which detects multiple sound events occurring simultaneously.

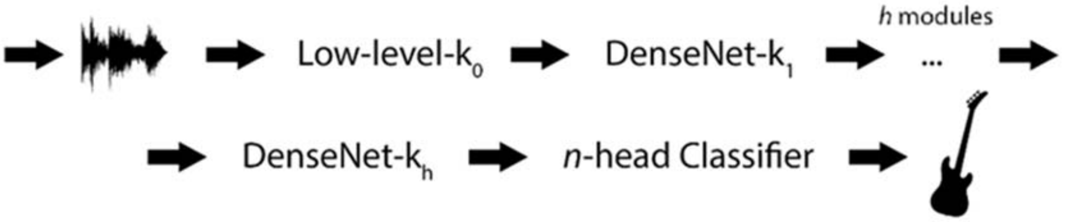


Fig. 1. Overall architecture for the model (k - denotes growth rate)

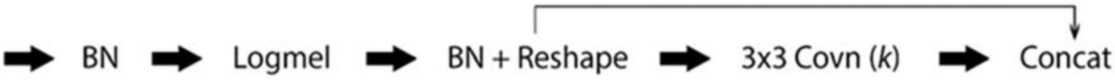


Fig. 2. Low-level-k module

In this work, we employ the same mixup settings as those utilized in the work of Il-Young Jeong and Hyungui Lim in the DCASE2018 challenge, as their approach resulted in the highest scores [27],[33]. We set λ to be a random variable of Beta(0.4, 0.4). To make the target class data predominate, we set a λ > 0.5 condition for the generated data. The class of the other data for the mixup is randomly selected. Finally, to random scale the data, we apply scale augmentation. The following equation represents this process [37]:

$$\hat{x} = \frac{w\lambda x_j}{\max(|x_j|)} + \frac{w(1-\lambda)x_k}{\max(|x_k|)} \quad (3)$$

where w is a random variable with uniform distribution for the scale augmentation. Incorporating mixup into the existing training scheme does not introduce a substantial computational overhead [37].

2 MODEL ARCHITECTURE

A mixup technique makes minimizing training data loss difficult, despite increasing validation and test accuracy. Therefore, the training efficiency against strong mixup augmentation is one of the main problems of the musical instrument classification strategy.

The full block diagram of the proposed model is presented in Fig. 1. It consists of a low-level feature-space module, dense neural network, and a classifier. They are presented in the subsequent subsections. Again, we followed the DCASE2018 paper to configure our NN.

2.1 LOW-LEVEL SIGNAL REPRESENTATION MODULE

It is worth noting that the dense neural network needs bigger representation presented at its input than a typical learning algorithm; therefore, various 2D feature spaces are employed, such as spectrograms, cepstrograms, chromagrams [4],[38],[39]. When working with spectral representations of audio, the mel-scale is a common reweighting of the frequency dimension, which results in a lower-dimensional and a more perceptually-relevant representation of the audio [40]. A logarithm of the mel-scale spectrogram is a widely used preprocessing step in audio signal analysis. Compressing the spectrogram magnitudes after reweighting the frequencies is different from reweighting the compressed spectrogram magnitudes. According to the perceptual justification of the mel-scale, conversion from the linear scale entails summing intensity or energy among adjacent bands, i.e., it should be applied before logarithmic compression. Taking the weighted sum of log-compressed values amounts to multiplying the pre-logarithm values, which rarely, if ever, makes sense. In this work, we applied log-mel transform as a low-level module in our model by using the Kapre script [41]. Kapre is an audio preprocessor (script) written in Python which can perform operations by using GPU instead of the physical memory, with the use of the cuDNN (NVIDIA CUDA Deep Neural Network) library. In general, this results in reducing memory consumption as well as the time needed to train NN.

A detailed low-level module is presented in Fig. 2. First, the input waveform is normalized by using Batch normalization (BN). Batch normalization is applied so that the distribution of the inputs (and these inputs are literally the result of an activation function) to a specific layer does not change over time, due to the parameter updates from each batch (or at least, allows it to change in an advantageous way). Then, it is transformed into a log-mel domain, which has two dimensions of time and frequency (see Fig. 3. in which an example of 2D mel-scale spectrogram is shown). After that, the input is reshaped to have the size of {time, frequency, 1}. In order to make the output features of the convolution layer concatenated with its inputs, we utilized a single-block, densely-connected architecture. In the presented low-level module, the k variable is used without an index because it is an example of only one module.

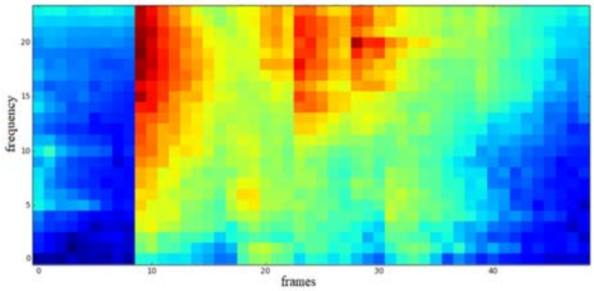


Fig. 3. An example of the mel-scale spectrogram of a cello sound

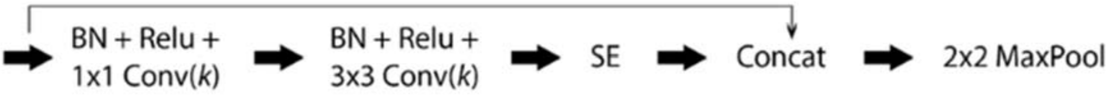


Fig. 4. DenseNet-k module

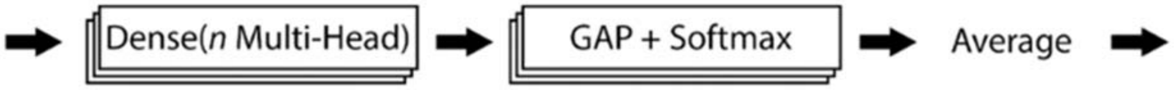


Fig. 5. n-head classifier module

2.2 DenseNet module

In the arXiv paper, Kilian Q. Weinberger and his team introduced direct connections from any layer to all subsequent layers and because of its dense connectivity they refer to this network architecture as Dense Convolutional Network (DenseNet) [29]. In this paper, only a single-block architecture is applied, so the very first log-mel can reach even the last layer. In some cases, the Neural Network in these layers could be "transparent", so no processing might be performed at all. It was discovered during the experiments, that the number of blocks determines the time of training; thus, an architecture consisting of one block shortens the training phase significantly.

The DenseNet module details are shown in Fig. 4. The filter size of the layer output increases after each iteration due to concatenation. We found out that reduction by using convolution solves this problem, so we are using a 1x1 convolution first, and then a 3x3 convolution [29]. Next, we applied the Squeeze-and-Excitation Network [42]. The Squeeze-and-Excitation Networks (SEs) introduce a building block for CNNs that improves channel interdependencies at almost no computational cost. SE adds a content-aware mechanism to weight each channel adaptively and is expected to help efficient training by adding only a few more parameters [29].

Another operation, namely pooling, helps in avoiding overfitting. There are two types of pooling operations that could be performed:

- Max Pooling – selecting the maximum value
- Mean Pooling – summing all of the values and dividing it by the total number of values

Mean Pooling is rarely used, so for the last layer of each module, a 2x2 max pooling is applied.

2.3 CLASSIFIER MODULE

In general, the goal of the classification task is to predict the target label, which is binary, e.g., [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]. In the case of our study, this refers to musical instruments being recognized. Therefore, the Neural Network should correctly classify musical instruments and return one binary label for each instrument. So, for the vector [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1], the classified label output would be, e.g., "cello", etc. Unfortunately, when the mixup is applied, it needs to predict the real values in the range of (0, 1), e.g., [0.1, 0, 0.9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] or [0.2, 0.1, 0.6, 0, 0.1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]. One of the downsides of applying mixup augmentation is that the target label values tend to be closer to 0.5 rather than to 1, which complicates the task of the classification. We modified the conventional softmax output layer so that it has a multi-head architecture (the output here is obtained by averaging multiple softmax outputs, as shown in Fig. 5). This helps to train the mixup augmentation models.

The softmax function is often used in the final layer of the NN-based classifier. Such a network is typically trained under a log-loss (or cross-entropy) measures, giving a non-linear variant of multinomial logistic regression [43]. In this experiment, instead of returning a binary target label, the output of the classifier is a probability for every class label, while hinge loss gives the max-margin. This type of loss penalizes wrong predictions significantly and to less extent - not confident predictions and the correct class score exceeds the other scores by more than the margin. The output of this softmax layer lies between 0 and 1.

3 TRAINING DATA AND OPTIMIZATION

The training procedure was performed by using Keras [44], which is an open-source Neural Network library written in the Python environment. AdamOptimizer class was used for optimization [45]. Although it adaptively controls the learning rate (*lr*) by itself, we found that manual decaying of the learning rate helps the optimization process. Validation accuracy was apprised for every 1000 iterations; next, the best model was saved for the analysis. The computation time for 1000 iterations was about 2 minutes (logmel-based model) and about 3 minutes (waveform-based model), using a PC with the Asus x99 motherboard, Intel i9 CPU, 32 Gb RAM, and N24kVIDIA 1080 GPU.

3.1 TRAINING DATA

For every instrument, we created a separate folder that contains a raw instrument signal recording from the Cambridge database [34]. As already mentioned, the analyzed songs belong to 20 music genres and are available in this database (see Table 1). From every multitrack available, a specific track was taken and automatically cut, employing scripts that use SOX [46] (a command-line audio processing tool for signal cutting and naming) and ffmpeg [47] (a script for silence detection in an audio stream). For example, in the Cambridge database, there is a multitrack from the band called Boogiesnakes. The song title is “It's My Right”, and it consists of 19 separate raw tracks (see Table 2). To create an even bigger training set, we cut every raw track using a Windows PowerShell [48] script that detects silence, cuts the edited track, and saves it. Whenever silence (longer than 500 ms) was detected, the cut was performed, and the file was saved with an appropriate name. The threshold for silence was set empirically, i.e., at -50 dB RMS, to allow cutting even on noisy tracks. That is why we were able to build a larger database which consists of 576484 samples. Table 1 contains the corresponding number of tracks utilized.

Music genre	No. of multitracks (original raw tracks)	No. of multitracks (processed tracks)
acoustic	23	15456
jazz	18	23184
country	7	7840
orchestral	7	6272
electronica	10	19040
dance	15	61455
experimental	3	2712
pop	40	103040
singer-songwriter	25	50400
alt rock	13	23296
blues	16	25088
country rock	6	8736
indie	18	36288
funk	14	18816
reggae	5	9120
rock	36	72072
punk	15	21840
metal	22	40656
hip-hop	10	21735
r&b	6	9438
altogether	4633	576484

Table 1. Music genres and the number of samples employed

Before cutting, there were 4633 tracks (raw downloaded recordings from the Cambridge database), so after this simple operation, our database grew significantly. Also, the original tracks were about 3 minutes long, which would slow down the process of training and validation extremely. It is very important to train a Neural Network with short (small) data snippets, because it speeds up the training, validating, and testing procedure. After this operation, the longest track is 56 seconds long. Recordings have not been normalized in any way. All tracks are 44.1 kHz/16 bit.

Training data contain bass, kick, snare (both top and bottom inside one folder), hi-hat, electric guitar, piano, electric keys, vocals (only main), high tom, medium tom, low (floor) tom, digital sub-bass, cymbals, violins, alto sax, and cello. Drums, bass, guitars, and vocals constitute the majority of available tracks (about 70% of the whole sample database). Other instruments were represented in a significantly smaller number of tracks. It is because every multitrack contains more than one instrument, e.g., the guitar track. Also, more often the silence occurs between separated hits on drums (kick, snare, and toms) than on string instruments. Fig. 6 shows an example of a spectrogram and a waveform of a kick drum with markers on places where cuts would be made.

3.2 BATCH-WISE LOSS MASKING

Another consideration in the optimization process is to label noise, since all recordings are real-life examples. In the Cambridge database, there are 19 separate folders for every instrument, each containing about 24000 samples in total. The labels of only 95000 items are verified, and it is not guaranteed that the remaining ones have the correct label assigned. Only for data named properly, e.g., “bass” or “hi-hat”, the label (vector) may be generated automatically. Others were labeled with some basic titles, e.g., “Track01”. Because it is impossible to listen to all of the recordings and correctly label the data manually, the model should be trained to handle those outliers. Moreover, falsely-labeled data may lead to lower classification performance and slow down the process of optimization. Based on the following operations, we expect to remove data with false labels automatically.

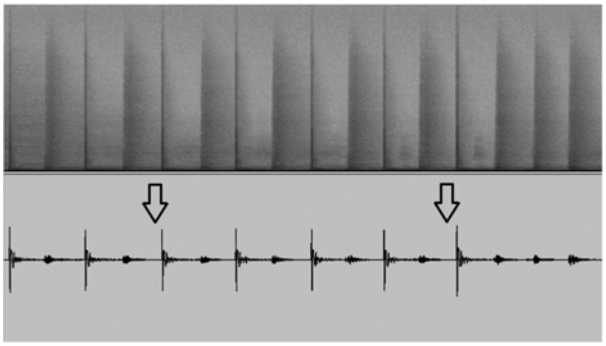


Fig. 7. Kick drum and correctly marked cut places

In this paper, we utilized the batch-wise loss masking approach. The conventional loss function for a mini-batch is defined as follows [26]:

$$J = \sum_n C_n \tag{4}$$

where C_n is cross-entropy for a single data in a mini-batch, defined by Eq. (5):

$$C_n = \sum_c t_{n,c} \log(y_{n,c}) \tag{5}$$

Under a condition that data labeled correctly are known, the loss function may be modified according to Eq. (6) to ignore noisy data:

$$\hat{J} = \sum_n m_n C_n \tag{6}$$

where m_n is 1 if n-th data item is labeled correctly and 0 if not.

Raw tracks	1	2	3	4	5	6	7	8	9
	Kick	Kick_Sub	Snare_Up	Snare_Down	Hihat	Tom1	Tom2	Overhead	Drum_Room
10	11	12	13	14	15	16	17	18	19
Bass_DI	Bass_mic	Gtr1_mic1	Gtr1_mic2	Gtr2_mic1	Gtr2_mic2	Gtr3_mic1	Gtr3_mic2	Gtr3_DI	Lead_Vocal

Table 2 Raw music tracks of the song “It's My Right” by Boogiesnakes

To decide the values of m, it should be checked whether this is a case of the verified data or not. If yes, then a true label is assigned. Contrarily, if some data show an especially high loss in the current model, then it can be considered as an outlier with a noisy-label [27].

$$m_n = \begin{cases} 1\ if\ v_n = 1\ or\ C_n < \mu \\ 0 \qquad\quad otherwise \end{cases} \tag{7}$$

where v_n denotes whether the n-th data item is manually verified or not; μ is defined as in the work of Il-Young and Hyungui [26]:

$$\mu = \alpha \times \max_n C_n \tag{8}$$

where α is empirically set to be 0.8 [27].

To calculate gradient, we used this modification, which eliminates data with the largest error. Moreover, in every batch, there is some chosen data that will be eliminated and will help to find more falsely-labeled data in next iterations. The described masking technique improved the cross-validation accuracy by about 0.99 percent point. With such a large amount of the training data, this is a very big profit.

4 RESULTS

The data were split according to the 60/20/20% rule. From the whole set, 60% of data from each class were used for the model training, 20% for the model validation, and the remaining 20% for testing. Figure 7 shows a confusion matrix containing results of classification of musical instruments based on test data.

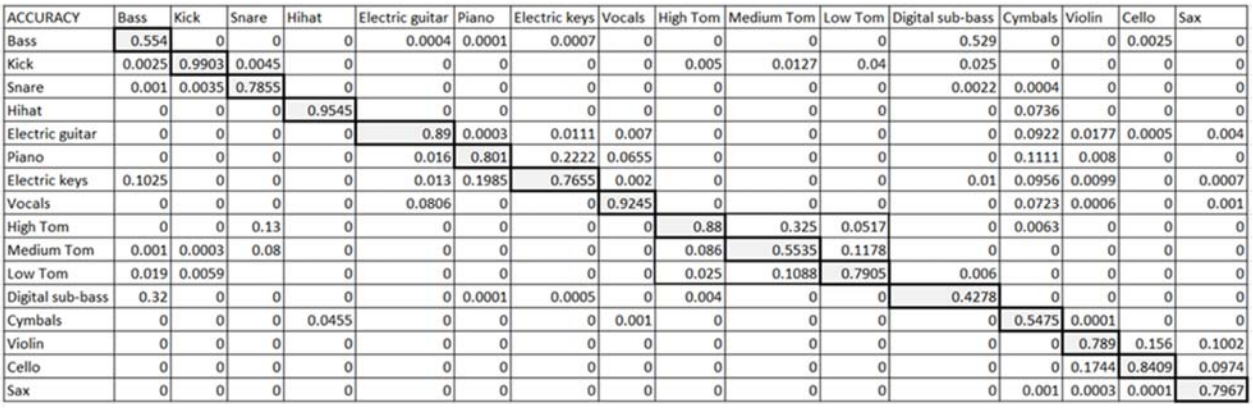
As seen from the confusion matrix, some of the instruments were classified rather poorly, e.g., medium tom – its recognition score was distributed throughout high and low tom, whereas the boundary cases (high and low tom) were classified correctly. We think the improper classification may be a result of the fact that the softmax classifier is followed by an averaging function (some of the results tend to be closer to 0.5 than others). Also, as said in the previous subsection, data could be classified (and trained) improperly due to their unknown labels.

In some cases, different instruments have very similar time- and frequency domain characteristics, e.g., envelopes, frequency responses, etc., what makes them difficult to discern. For example, we introduced digital sub-bass to NN along with acoustic bass, and in both cases, accuracies of the recognition process were roughly equal, and samples were mislabeled in these two classes. In fact, frequency ranges of these samples overlap to some extent; thus, instrument similarity causes this problem. Moreover, when the state-of-the-art VSTIs are used in popular music, it is very difficult to recognize whether an instrument is a real recording or just a digital creation. Another case of misclassification can be observed for low, medium, and high toms. So, if we group similar instruments, e.g., digital sub-bass and bass as “bass”, we may acquire higher recognition rates.

In raw recordings of kick, there is a bleed from other drums (mostly snare) and vice versa. This made the task of correctly classifying these instruments harder. It was impossible to edit those tracks or to use some dynamic operations on them, such as gating, because of their amount and characteristics that require different settings for each track.

5 CONCLUSIONS

In this paper, we presented a promising method for automatic music instrument tagging system using Neural



ACCURACY	Bass	Kick	Snare	Hihat	Electric guitar	Piano	Electric keys	Vocals	High Tom	Medium Tom	Low Tom	Digital sub-bass	Cymbals	Violin	Cello	Sax
Bass	0.554	0	0	0	0.0004	0.0001	0.0007	0	0	0	0	0.529	0	0	0.0025	0
Kick	0.0025	0.9903	0.0045	0	0	0	0	0	0.005	0.0127	0.04	0.025	0	0	0	0
Snare	0.001	0.0035	0.7855	0	0	0	0	0	0	0	0	0.0022	0.0004	0	0	0
Hihat	0	0	0	0.9545	0	0	0	0	0	0	0	0	0.0736	0	0	0
Electric guitar	0	0	0	0	0.89	0.0003	0.0111	0.007	0	0	0	0	0.0922	0.0177	0.0005	0.004
Piano	0	0	0	0	0.016	0.801	0.2222	0.0655	0	0	0	0	0.1111	0.008	0	0
Electric keys	0.1025	0	0	0	0.013	0.1985	0.7655	0.002	0	0	0	0.01	0.0956	0.0099	0	0.0007
Vocals	0	0	0	0	0.0806	0	0	0.9245	0	0	0	0	0.0723	0.0006	0	0.001
High Tom	0	0	0.13	0	0	0	0	0	0.88	0.325	0.0517	0	0.0063	0	0	0
Medium Tom	0.001	0.0003	0.08	0	0	0	0	0	0.086	0.5535	0.1178	0	0	0	0	0
Low Tom	0.019	0.0059		0	0	0	0	0	0.025	0.1088	0.7905	0.006	0	0	0	0
Digital sub-bass	0.32	0	0	0	0	0.0001	0.0005	0	0.004	0	0	0.4278	0	0	0	0
Cymbals	0	0	0	0.0455	0	0	0	0.001	0	0	0	0	0.5475	0.0001	0	0
Violin	0	0	0	0	0	0	0	0	0	0	0	0	0	0.789	0.156	0.1002
Cello	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1744	0.8409	0.0974
Sax	0	0	0	0	0	0	0	0	0	0	0	0	0.001	0.0003	0.0001	0.7967

Fig. 7. Confusion matrix containing musical instrument classification results

Networks. The method proved to provide sufficiently high recognition rates, even with recordings from real-life environments that contain noisy data. We focused primarily on how to efficiently train the model for every instrument to not only obtain the highest result, but also to speed up the process of validation. This paper shows that Neural Network constructed for one task can also be trained on different data and can return high classification rates.

In the future work, evaluation based on a larger number of musical instruments, including more synthetic instruments, i.e., synths, will be performed. Also, differentiated musical articulation will be taken into account. Another area that should be checked in the future work is defining a better model that recognizes similar instruments such as digital and acoustic bass, or low and medium tom. Moreover, an automatic mixing system will be proposed for which the created tagging system will be applied.

REFERENCES

[1] O. Celma, P. Herrera, X. Serra, Bridging the Music Semantic Gap, Workshop on Mastering the Gap: From Information Extraction to Semantic Representation, Budva, Montenegro, (2006).

[2] B. Kostek, Musical Instrument Classification and Duet Analysis Employing Music Information Retrieval Techniques, May 2004, Proceedings of the IEEE 92(4):712 – 729. https://doi.org/10.1109/jproc.2004.825903.

[3] B. Kostek, Perception-Based Data Processing in Acoustics. Applications to Music Information Retrieval and Psychophysiology of Hearing, Springer Verlag, Series on Cognitive Technologies, Berlin, Heidelberg, New York, 2005.

[4] G. Korvel, P. Treigys, G. Tamulevicius, J. Bernataviciene, B. Kostek, Analysis of 2D Feature Spaces for Deep Learning-Based Speech Recognition, J. Audio Eng. Soc., vol. 66, no. 12, pp. 1–10, (2018 November.), DOI: https://doi.org/10.17743/jaes.2018.0066.

[5] G. Tzanetakis, G. Essl, P. Cook, Automatic musical genre classification of audio signals, in Proc. Int.Symp. Music Information Retrieval (ISMIR), Oct. 2001.

[6] DeLiang Wang, G.J. Brown (Editors), Computational Auditory Scene Analysis: Principles, Algorithms, and Applications, Wiley-IEEE, A JOHN WILEY & SONS, INC., PUBLICATION, 2006.

[7] J. Zeremdini, M.A. Ben Messaoud, A. Bouzid, A comparison of several computational auditory scene analysis (CASA) techniques for monaural speech segregation, Brain Inform., 2(3), pp. 155-166, 2015. https://doi.org/10.1007/s40708-015-0016-0.

[8] J. A. Burgoyne, I. Fujinaga, J.S. Downie, Music Information Retrieval, Chapter 15 in A New Companion to Digital Humanities (Editors: S. Schreibman, R. Siemens, J. Unsworth, John Wiley & Sons, Ltd., 2016, doi: https://doi.org/10.1002/9781118680605.ch15.

[9] L. Vrysis, N. Tsipas, C. Dimoulas, G. Papanikolaou, Crowdsourcing Audio Semantics by Means of Hybrid Bimodal Segmentation with Hierarchical Classification, December 2016, J. Audio Eng. Soc., vol. 64, 12 pp. 1042-1054. https://doi.org/10.17743/jaes.2016.0051.

[10] M. Casey, A. Westner, Separation of mixed audio sources by independent subspace analysis. Proceedings of International Computer Music Conference, 154-161, Berlin, 2000.

[11] M. Dziubiński, P. Dalka, B. Kostek, Estimation of Musical Sound Separation Algorithm Effectiveness Employing Neural Networks, J. Intelligent Information Systems, vol. 24, 2, 133-157, 2005.

[12] A. Eronen, Musical instrument recognition using ICA-based transform of features and discriminatively trained HMMs, Proc. of the Seventh International Symposium on Signal Processing and its Applications, ISSPA 2003, Paris, France, 1-4 July 2003, pp. 133-136.

[13] A. Eronen, A. Klapuri, Musical Instrument Recognition Using Cepstral Coefficients and Temporal Features, Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2000, pp. 753-756.

[14] O. Gillet, G. Richard, Transcription and separation of drum signals from polyphonic music, IEEE Transactions on Audio, Speech and Language Processing, vol. 16, 529–540, 2008.

[15] P. Herrera, X. Amatriain, E. Batlle, X. Serra, Towards instrument segmentation for music content description: a critical review of instrument classification techniques, Proceedings of International Symp. on Music Information Retrieval, Plymouth, Massachusetts, 2000.

[16] T.C. Nagavi1, N.U. Bhajantri, An Extensive Analysis of Query by Singing/Humming System through Query Proportion, The International Journal of Multimedia & Its Applications (IJMA) vol. 4, 6, December 2012, pp. 73-86. https://doi.org/10.5121/ijma.2012.4606.

[17] B. Kostek, A. Czyżewski, Representing Musical Instrument Sounds for Their Automatic Classification, J. Audio Eng. Soc., vol. 49, 9, 768-785, 2001.

[18] B. Kostek, Soft Computing in Acoustics, Applications of Neural Networks, Fuzzy Logic and Rough Sets to Musical Acoustics, Studies in Fuzziness and Soft Computing, Physica Verlag, Heidelberg, New York 1999.

[19] R. Kotsakis, G. Kalliris, C. Dimoulas, Investigation of broadcast-audio semantic analysis scenarios employing radio-programme-adaptive pattern classification, Speech Communication, Vol. 54, No. 6, pp. 743-762,

2012. https://doi.org/10.1016/j.specom.2012.01.004.

[20] R. Kotsakis, G. Kalliris, C. Dimoulas, Investigation of salient audio-features for pattern-based semantic content analysis of radio productions, in Proceedings of the 132nd Audio Engineering Society (AES) Convention, paper no. 8663, Budapest, Hungary, April 2012. Permalink: http://www.aes.org/e-lib/browse.cfm?elib=16301 (AES E-Library)

[21] MPEG 7 standard, https://mpeg.chiariglione.org/standards/mpeg-7.

[22] G. De Poli, P. Prandoni, Sonological Models for Timbre Characterization, *Journal of New Music Research*, Vol 26 (1997), pp. 170-197, 1997.

[23] G. de Poli, P. Prandoni, Sonological Models for Timbre Characterization, Journal of New Music Research, Vol 26 (1997), pp. 170-197, 1997.

[24] G. de Poli, A. Piccialli, C. Roads (Editors), Representations of Musical Signals, MIT Press, Cambridge, MA, 1991.

[25] C. Papaodysseus, G. Roussopoulos, D. Fragoulis, Th. Panagopoulos, C. Alexiou, A New Approach to the Automatic Recognition of Musical Recordings, J. Audio Eng. Soc., Vol. 49, No. 1/2, 2001.

[26] A. Rosner, B. Kostek, Automatic music genre classification based on musical instrument track separation, J. Intell. Inf. Syst., vol. 50, 2, pp. 363-384. https://doi.org/10.1007/s10844-017-0464-5.

[27] J. Il-Young, L. Hyungui, Audio tagging system for dcase 2018: focusing on label noise, data augmentation and its efficient learning, Detection and Classification of Acoustic Scenes and Events, 2018.

[28] DCASE 2018 challenge, http://dcase.community/challenge2018/task-general-purpose-audio-tagging-results, 2018.

[29] G. Huang, Z. Liu, L. van der Maaten, Densely Connected Convolutional Networks, 2018.

[30] P. Ruiz, Understanding and visualizing DenseNets, http://www.pabloruizruiz10.com/resources/CNNs/DenseNets.pdf (accessed March 2019).

[31] J.T. Springenberg, A. Dosovitskiy, T. Brox, M. Riedmiller, Striving for simplicity: The all convolutional net. arXiv preprint arXiv:1412.6806, 2014.

[32] Y. Zhang, K. Lee, H. Lee, Augmenting supervised neural networks with unsupervised objectives for large-scale image classification. In ICML, 2016.

[33] D. Liang, F. Yang, T. Zhang, P. Yang, Understanding Mixup Training Methods. IEEE Access. PP. 1-1. October 2018. https://doi.org/10.1109/ACCESS.2018.2872698.

[34] Cambridge Music Technology website, http://www.cambridge-mt.com/ms-mtk.htm, 2019.

[35] P. Simard, Y. LeCun, J. Denker, B. Victorri, Transformation invariance in pattern recognitiontangent distance and tangent propagation. Neural networks: tricks of the trade, 1998.

[36] O. Chapelle, J. Weston, L. Bottou, V. Vapnik, Vicinal risk minimization. NIPS, 2000.

[37] H. Zhang, M. Cisse, Y. Dauphin, D. Lopez-Paz, mixup: Beyond Empirical Risk Minimization, arXiv: 1710:09412v2, 2018.

[38] A.M. Badshah, J. Ahmad, N. Rahim, S.W. Baik, Speech Emotion Recognition from Spectrograms with Deep Convolutional Neural Network, Intern. Conf. on Platform Technology and Service (PlatCon), pp. 1-5 (2017). https://doi.org/10.1109/PlatCon.2017.7883728.

[39] T. Ozseven, Investigation of the Effect of Spectrogram Images and Different Texture Analysis Methods on Speech Emotion Recognition, Applied Acoustics, vol. 142, 70–77 (2018). https://doi.org/10.1016/j.apacoust.2018.08.003.

[40] P. Mermelstein, Distance Measures for Speech Recognition, Psychological and Instrumental, Pattern Recognition and Artificial Intelligence, vol. 116, pp. 374–388 (1976).

[41] K. Choi, D. Joo, J. Kim, Kapre: On-gpu audio preprocessing layers for a quick implementation of deep neural network models with keras, arXiv preprint arXiv: 1706.05781, 2017.

[42] J. Hu, L. Shen, G. Sun, Squeeze-andexcitation networks, arXiv preprint arXiv: 1709.01507, vol. 7, 2017.

[43] B. Cheng, R. Xiao, Y. Guo, Y. Hu, J. Wang, L. Zhang, Revisit Multinomial Logistic Regression in Deep Learning: Data Dependent Model Initialization for Image Recognition, Computer Science-Computer Vision and Pattern Recognition, Sep 2018, arXiv.org > cs > arXiv:1809.06131.

[44] F. Chollet, Keras, https://keras.io, 2015.

[45] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv: 1412.6980, 2014.

[46] SOX command-line audio processing tool script, http://sox.sourceforge.net/sox.pdf, 2019.

[47] ffmpeg script for silence detection in audio stream, https://ffmpeg.org/ffmpeg.html, 2019.

[48] Windows PowerShell script, https://docs.microsoft.com/en-us/powershell/, 2019.

THE AUTHORS

Damian Koszewski

Bozena Kostek

D. Koszewski was born in 1991 in Gdansk. In 2017 he graduated from the Faculty of Electronics, Telecommunications and Informatics at Gdansk University of Technology (GUT) acquiring M.Sc. degree. The subject of his M.S. thesis was associated with subjective tests aimed at the assessment of sound similarity of the virtual and real musical instruments. He is a member of the Audio Engineering Society since 2014. Currently he works on his Ph.D. in GUT. His scientific interests concern sound and video processing and perception. His other interests include production and composition of music and playing musical instruments.

●

B. Kostek holds professorship at the Faculty of Electronics, Telecommunications and Informatics, Gdansk University of Technology (GUT), Poland. She is Head of the Audio Acoustics Laboratory. In 2013 she has been elected as a member of the Polish Academy of Sciences. She received her M.Sc. degrees in sound engineering (1983) and organization and management (1986) from GUT. She also received postgraduate DEA degree (1988) from Toulouse University, France. In 1992 she supported her Ph.D. thesis with honors at GUT, and in 2000 her D.Sc. degree at the Research Systems Institute, Polish Academy of Sciences. In 2005 the President of Poland granted her the title of Professor. She published over 600 scientific papers in journals and at international conferences. Under her guidance, 14 Ph.D. students supported their doctoral theses and she supervised over 260 M.Sc. and Eng. Theses. She serves as the Editor-in-chief of the *Journal of the Audio Engineering Society* since 2011. She was also the Editor-in-Chief of *Archives of Acoustics* (2007-2012). She was the recipient of many prestigious awards for research, including those of the Prime Minister of Poland (2000, 2014) for outstanding research achievements, prizes of the Polish Academy of Sciences and Ministry of Science and the Bachelor Cross of the Polonia Restituta Order (2011). She also received the Audio Engineering Society Fellowship Award in 2010, and the AES Citation Award in 2013. Her research activities are interdisciplinary, however the main research interests focus on audio signal processing, music informatics, human-computer interaction, cognitive bases of sound and vision processing, as well as psychophysiology of hearing and vision.