DATA AUGMENTATION FOR INSTRUMENT CLASSIFICATION ROBUST TO AUDIO EFFECTS

António Ramires

Music Technology Group
Universitat Pompeu Fabra
Barcelona, Spain
antonio.ramires@upf.edu

Xavier Serra

Music Technology Group
Universitat Pompeu Fabra
Barcelona, Spain
xavier.serra@upf.edu

ABSTRACT

Reusing recorded sounds (sampling) is a key component in Electronic Music Production (EMP), which has been present since its early days and is at the core of genres like hip-hop or jungle. Commercial and non-commercial services allow users to obtain collections of sounds (sample packs) to reuse in their compositions. Automatic classification of one-shot instrumental sounds allows automatically categorising the sounds contained in these collections, allowing easier navigation and better characterisation.

Automatic instrument classification has mostly targeted the classification of unprocessed isolated instrumental sounds or detecting predominant instruments in mixed music tracks. For this classification to be useful in audio databases for EMP, it has to be robust to the audio effects applied to unprocessed sounds.

In this paper we evaluate how a state of the art model trained with a large dataset of one-shot instrumental sounds performs when classifying instruments processed with audio effects. In order to evaluate the robustness of the model, we use data augmentation with audio effects and evaluate how each effect influences the classification accuracy.

1. INTRODUCTION

The repurposing of audio material, also known as sampling, has been a key component in Electronic Music Production (EMP) since its early days and became a practice which had a major influence in a large variety of musical genres. The availability of software such as Digital Audio Workstations, together with the audio sharing possibilities offered with the internet and cloud storage technologies, led to a variety of online audio sharing or sample library platforms. In order to allow for easier sample navigation, commercial databases such as sounds.com[1] or Loopcloud[2] rely on expert annotation to classify and characterise the content they provide. In the case of collaborative databases such as Freesound [1] the navigation and characterisation of the sounds is based on unrestricted textual descriptions and tags of the sounds provided by users. This leads to a search based on noisy labels which different members use to characterise the same type of sounds.

Automatically classifying one-shot instrumental sounds in unstructured large audio databases provides an intuitive way of navigating them, and a better characterisation the sounds contained. For databases where the annotation of the sounds is done manually, it can be a way to simplify the job of the annotator, by providing suggested annotations or, if the system is reliable enough, only presenting sounds with low classification confidence.

The automatic classification of one-shot instrumental sounds remain an open research topic for music information retrieval (MIR). While the research on this field has been mostly performed on clean and unprocessed sounds, the sounds provided by EMP databases may also contain "production-ready" sounds, with audio effects applied on them. Therefore, in order for this automatic classification to be reliable for EMP sample databases, it has to be robust to the types of audio effects applied to these instruments. In our study, we evaluate the robustness of a state of the art automatic classification method for sounds with audio effects, and analyse how data augmentation can be used to improve classification accuracy.

2. RELATED WORK

Automatic instrument classification can be split into two related tasks with a similar goal. The first is the identification of instruments in single instrument recordings (which can be isolated or overlapping notes) while the second is the recognition of the predominant instrument in a mixture of sounds. A thorough description of this task and an overview of the early methodologies used is presented in [2]. These early approaches used two modules for classification, one for extracting and selecting handcrafted features (e.g. Mel Frequency Cepstral Coefficients, spectral centroid, roll-off, and flux) and another for classification (e.g. k-nearest neighbours, support vector machines or hidden Markov models). Datasets used for the evaluation and training of these algorithms included RWC [3] or the University of Iowa Musical Instrument Samples[3]. While these datasets are small (RWC has 50 instruments) they proved to be good for classification using handcrafted features. New datasets such as IRMAS [4] for predominant instrument classification and GoodSounds [5] with single instrument recordings have been created and provided sufficient data for deep learning approaches to be able to surpass more traditional machine learning approaches. A review of the evolution of traditional machine learning and deep learning approaches for instrument classification is presented in [6]. While the performance of traditional machine learning methods rely on developing handcrafted features, deep learning methods learn high-level representations from data using a general-purpose learning procedure, eliminating the need of expert feature extraction [7]. However, the success of these approaches is highly dependent on both the type

[1]https://sounds.com/

[2]https://www.loopcloud.net/

[3]http://theremin.music.uiowa.edu/MIS.html

and amount of data they are provided [8].

Recent work has shown the effectiveness of using Convolutional Neural Networks (CNNs) for instrument classification [9–12]. CNNs can be seen as trainable feature extractors, where kernels (or filters) with trainable parameters are convolved over an input, being able to capture local spatial and temporal characteristics. This architecture has been applied with great success to the detection, segmentation and recognition of objects and regions in images [7]. In the audio domain, when raw audio or spectograms are given, CNNs are able to learn and identify local spectro-temporal patterns relevant to the task to which they are applied. When utilized for MIR tasks, CNNs have outperformed previous state of the art approaches for various tasks [10, 13]. For automatic instrument classification, the state of the art approaches use CNNs trained on different representations of the input, such as raw audio [11], spectograms together with multiresolution recurrence plots [12] and log mel-frequency spectograms [9, 10]. In [9], CNNs were tailored towards learning timbre representations in log mel-frequency spectograms through the use of vertical filters instead of the commonly used square filters. For instrument classification, this approach displays a close to the state of the art [10] accuracy on the IRMAS dataset [4], while reducing the number of trainable parameters by approximately 23 times, on the single-layer proposed model.

Within the context of NSynth [14], a new high-quality dataset of one shot instrumental notes was presented, largely surpassing the size of the previous datasets, containing 305979 musical notes with unique pitch, timbre and envelope. The sounds were collected from 1006 instruments from commercial sample libraries and are annotated based on their source (acoustic, electronic or synthetic), instrument family and sonic qualities. The instrument families used in the annotation are bass, brass, flute, guitar, keyboard, mallet, organ, reed, string, synth lead and vocal. The dataset is available online[4] and provides a good basis for training and evaluating one shot instrumental sound classifiers. This dataset is already split in training, validation and test set, where the instruments present in the training set do not overlap with the ones present in validation and test sets. However, to the best of our knowledge, no methods for instrument classification have so far been evaluated on this dataset.

In order to increase the generalisation of a model further than the data provided to it, one possible approach is to use data augmentation. This approach can be described as applying deformations to a collection of training samples, in a way that the correct labels can still be deduced [15]. In computer vision, transforming images by cropping, rotation, reflection or scaling are commonly used techniques for data augmentation. In the audio domain, an intuitive and practical transformation is applying audio effects to the original training audio files. Transformations such as time-stretching, pitch-shifting, dynamic range compression and adding background noise have been applied with success to environmental sound classification, for overcoming the data scarcity problems [16]. In [17], artificial reverberation was applied to speech recordings, so as to create a speech recognition system robust to reverberant speech. For instrument recognition, the same set of effects used in [16] was applied with success in [15]. We believe that the use of audio effects typically used in EMP such as echo, reverb, chorus, saturation, heavy distortion or flanger can lead to a useful augmentation, as well as to an increase in robustness in instrument classification scenarios where the instrument recordings have these effects applied.

3. METHODOLOGY

In our study we will conduct two experiments. First, we will try to understand how augmenting a dataset with specific effects can improve instrument classification and secondly, we will see if this augmentation can improve the robustness of a model to the selected effect.

To investigate this, we process the training, validation and test sets of the NSynth [14] dataset with audio effects. A state of the art deep learning architecture for instrument classification [9] is then trained with the original training set, and appended with each of the augmented datasets for each effect. We use the model trained with the original training set as a baseline and compare how the models trained with augmented versions perform on the original test and on the augmented versions of it for each effect. The code for the experiments and evaluation is available in a public GitHub repository[5].

3.1. Data Augmentation and Pre-Processing

The audio effects for the augmentation were applied directly to the audio files present in the training, validation splits of the NSynth dataset [14]. For the augmentation procedure, we used a pitch-shifting effect present in the LibROSA[6] library and audio effects in the form of VST audio plugins. For the augmentation which used audio plugins, the effects were applied directly to the audio signals using the Mrs. Watson[7] command-line audio plugin host. This command line tool was designed for automating audio processing tasks and allows the loading of an input sound file, processing it using a VST audio effect and saving the processed sound. In order to maintain transparency and reproducibility of this study only VST plugins which are freely distributed online were selected. The parameters used in the augmentation procedure were the ones set in the factory default preset for each audio plugin, except for those whose default preset did not alter significantly the sound.

The audio effects used were the following:

- **Heavy distortion:** A Bitcrusher audio effect which produces distortion through the reduction of the sampling rate and the bit depth of the input sound was used in the training set. The VST plugin used for augmenting the training set was the TAL-Bitcrusher[8]. For the test and validation set, we used Camel Audio's CamelCrusher[9] plugin which provides distortion using tube overdrive emulation combined with a compressor.
- **Saturation:** For this effect, tube saturation and amplifier simulation plugins were used. The audio effect creates harmonics in the signal, replicating the saturation effect from a valve- or vacuum-tube amplifier [18]. For this augmentation we focused on a subtle saturation which did not create noticeable distortion. The plugin used in the training set was the TAL-Tube[8], while for the validation and test set

[4]`https://magenta.tensorflow.org/datasets/nsynth`

[5]`https://github.com/aframires/instrument-classifier/`

[6]`https://librosa.github.io/librosa/`

[7]`https://github.com/teragonaudio/MrsWatson`

[8] `https://tal-software.com/products/tal-effects`

[9]`https://www.kvraudio.com/product/camelcrusher-`

Shattered Glass Audio's Ace[10] replica of a 1950s all tube amplifier was used.

- **Reverb:** To create a reverberation effect, the TAL-Reverb-4 plugin[11] was used in the test set. This effect replicates the artificial reverb obtained in a plate reverb unit. For the validation and test set we used OrilRiver[12] algorithmic reverb, which models the reverb provided by room acoustics. The default preset for this plugin mimics the reverb present in a small room.
- **Echo:** A delay effect with long decay and with a big delay time (more than 50ms) [18] was used to create an echo effect. We used the TAL-Dub-2[13] VST plugin in the training set and soundhack's ++delay[14] validation and test set. For this last plugin, we adapted the factory default preset, changing the delay time to 181.7 ms and the feedback parameter to 50%, so that the echo effect was more noticeable.
- **Flanger:** For this delay effect, the input audio is summed with a delayed version of it, creating a comb filter effect. The time of the delay is short (less than 15 ms) and is varied with a low frequency oscillator [18, 19]. Flanger effects can also have a feedback parameter, where the output of the delay line is routed back to its input. For the training set, the VST plugin used was the TAL-Flanger[8], while for the test and validation sets we used Blue Cat's Flanger[15], which mimics a vintage flanger effect.
- **Chorus:** The chorus effect simulates the timing and pitch variations present when several individual sounds with similar pitch and timbre play in unison [19]. The implementation of this effect is similar to the flanger. The chorus uses longer delay times (around 30 ms), a larger number of voices (more than one) and normally does not contain the feedback parameter [18, 19]. The VST effect used in the training set was the TAL-Chorus-LX[16] which tries to emulate the chorus module present in the Juno 60 synthesizer. For the test and validation sets, we used Blue Cat's Chorus[17], which replicates a single voice vintage chorus effect.
- **Pitch shifting:** For this effect, the LibROSA Python package for musical and audio analysis was used. This library contains a function which pitch shifts the input audio. As the dataset used contains recordings of the instruments for every note in the chromatic scale in successive octaves, our approach focused on pitch-shifting in steps smaller than one semitone, similarly to what can occur in a detuned instrument. The `bins_per_octave` parameter of the pitch-shifting function was set to $72 = 12 \times 6$ while the `n_steps` parameter was set to a random value between 1 and 5 for each sound. Neither 0 or 6 were selected as possible values as it would be the same as not altering the sound or pitch-shifting it by one semitone. The intention of the random assignment in the `n_steps` is to ensure the size of this augmented dataset is equal to the size of the datasets of other effects.

[10]`http://www.shatteredglassaudio.com/product/103`

[11]`https://tal-software.com/products/tal-reverb-4`

[12]`https://www.kvraudio.com/product/orilriver-by-denis-tihanov`

[13]`https://tal-software.com/products/tal-dub`

[14]`http://www.soundhack.com/freeware/`

[15]`https://www.bluecataudio.com/Products/Product_Flanger/`

[16]`https://tal-software.com/products/tal-chorus-lx`

[17]`https://www.bluecataudio.com/Products/Product_Chorus`

The audio resulting from this augmentation step can be longer than the original unprocessed audio. In order to keep all examples with the same length, the processed audio files were trimmed, ensuring all audio samples had a fixed duration of 4 s, similar to the sounds presented in the NSynth dataset [14].

The next step in the data processing pipeline is representing each sound in a log-scaled mel-spectogram. First, a 1024-point Short-time Fourier transform (STFT) is calculated on the signal, with a 75% overlap. The magnitude of the STFT result is converted to a mel-spectogram with 80 components, covering a frequency range from 40 Hz to 7600 Hz. Finally, the logarithm of the mel-spectogram is calculated, resulting in a 80×247 log-scaled mel-spectogram for the 4 s sounds sampled at 16 kHz present in the NSynth dataset [14].

3.2. Convolutional Neural Network

The CNN architecture we chose to use in our experiment is the *single-layer* architecture proposed by Pons et al. [9] for the musical instrument classification experiment, which has an implementation available online[18]. This architecture uses vertical convolution filters in order to better model timbral characteristics present in the spectogram, achieving close to state-of-the-art results [10], using a much smaller model (23 times less trainable parameters) and a consequently lower training time.

We chose the *single-layer* architecture presented in this study and adapted it to take an input of size 80×247. This architecture contains a single but wide convolutional layer with different filters with various sizes, to capture the timbral characteristics of the input:

- 128 filters of size 5×1 and 8×1;
- 64 filters of size 5×3 and 80×3;
- 32 filters of size 5×5 and 80×5.

Batch normalisation [20] is used after the convolutional layer and the activation function used is Exponential Linear Unit [21]. Max pooling is applied in the channel dimension for learning pitch invariant representations. Finally, 50% dropout is applied to the output layer, which is a densely connected 11-way layer, with the *softmax* activation function. A graph of the model can be seen in Figure 1. For more information on this architecture and its properties see [9].

3.3. Evaluation

The training of the models used the Adam optimiser [22], with a learning rate of 0.001. In the original paper [9] the authors used Stochastic Gradient Descent (SGD) with a learning rate reduction every 5 epochs. This was shown to provide good accuracy on the IRMAS dataset. However, we chose to use Adam as an optimiser because it does not need significant tuning as SGD. Furthermore, using a variable learning rate dependent on the number of epochs could benefit the larger training datasets as is the case of the ones with augmentation. A batch size of 50 examples was used, as it was the largest batch size able to fit the memory of the available

[18]`https://github.com/Veleslavia/EUSIPCO2017`

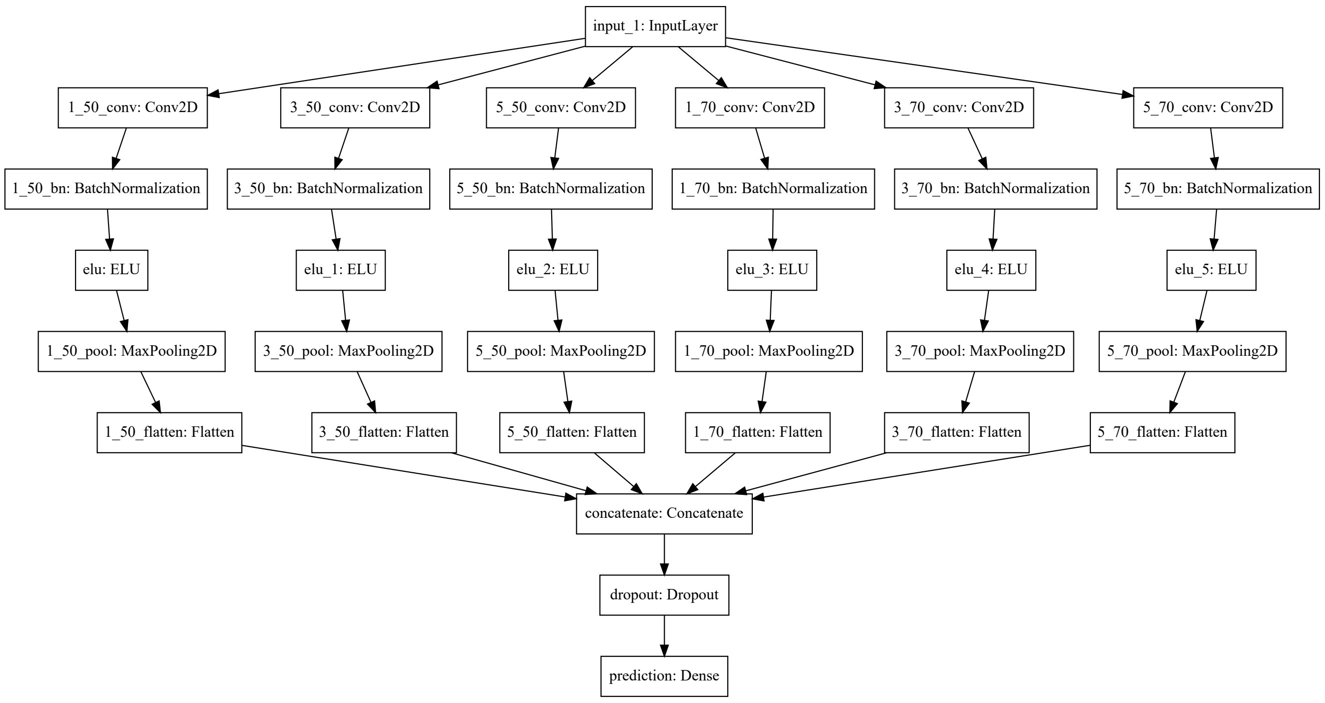

Figure 1: Single-layer CNN architecture proposed in [9]

GPUs. The loss function employed for the training was the categorical cross-entropy, as used in [9], which can be calculated as shown in Equation (1), where N represents the number of observations (examples in the training set) and $p_{model}[y_i \in C_{y_i}]$ is the predicted probability of the i^{th} observation belonging to the correct class C_{y_i}.

$$loss = -\frac{1}{N}\sum_{i=1}^{N}\log p_{model}[y_i \in C_{y_i}] \tag{1}$$

To compare the models trained with the different datasets, we used categorical accuracy as evaluation metric, described in Equation (2). A prediction is considered correct if the index of the output node with highest value is the same as the correct label.

$$\text{Categorical Accuracy} = \text{Correct predictions}/\text{N} \tag{2}$$

All the models were trained until the categorical accuracy did not improve in the validation set after 10 epochs and the model which provided the best value for the validation set was evaluated in the test set.

4. RESULTS

Two experiments were conducted in our study. We firstly evaluated how augmenting the training set of NSynth [14] by applying audio effects to the sounds can improve the automatic classification on the instruments of the unmodified test set. In the second experiment we evaluated how robust a state of the art model for instrument classification is when classifying sounds where these audio effects are applied.

The results of the first experiment are presented in Table 1, where the classification accuracy between the models trained with the original NSynth training set augmented with audio effects can be compared to the baseline (unprocessed dataset). We see that the increase in accuracy only occurs for chorus, heavy distortion and flanger effects. The highest classification accuracy was achieved by the dataset augmented with heavy distortion, where an increase of 1% was obtained. However, all the accuracy values are in a small interval (between 0.7334 and 0.7473), which means that the model was not able to learn from the augmented datasets. Future experiments are needed in order to understand why this occurs. In [16], the authors state that the superior performance obtained was due to an augmentation procedure coupled with an increase in the model capacity. Experiments with higher capacity models will be performed to understand if the size of the model used is limiting its performance on learning from the augmented dataset.

Table 1: Classification accuracy on the unprocessed test set.

Test Effect	Train Effect	Accuracy
None	None (baseline)	0.7378
	Heavy distortion	**0.7473**
	Saturation	0.7349
	Reverb	0.7375
	Chorus	**0.7417**
	Echo	0.7336
	Flanger	**0.7412**
	Pitch Shifting	0.7334

In Table 2, we present the accuracy values obtained when evaluating the trained model on test sets processed with effects. The first thing we verify is that the accuracy of the classification greatly decreases for almost all effects, when compared to the un-

Table 2: Classification accuracy on the augmented test set.

Test Effect	Train Effect	Accuracy
Heavy distortion	None	0.3145
	Heavy distortion	**0.3518**
Saturation	None	**0.4836**
	Saturation	0.4607
Reverb	None	**0.3931**
	Reverb	0.3774
Chorus	None	0.6348
	Chorus	**0.6436**
Echo	None	**0.4719**
	Echo	0.4319
Flanger	None	**0.7046**
	Flanger	0.7002
Pitch Shifting	None	**0.6980**
	Pitch Shifting	0.6741

processed sound classification. The model seems to be more robust to the flanger and to the pitch shifting effect, where the difference between the accuracy on the unprocessed test set and on the processed one is smaller than 4%. The effects which caused the biggest drops in accuracy ($> 20\%$) were the heavy distortion, the saturation, the echo and the reverb. When evaluating if training with the augmented datasets increased the robustness of the model, we see that this is only true for the chorus and distortion effect. While for the heavy distortion effect the accuracy when training with the augmented set is improved by a significant value ($\approx 4\%$), the difference in accuracy between training with the augmented and the unprocessed sets are small. Further experiments will be performed to understand the bad generalisation of the model. Besides experimenting with a higher capacity model as previously stated, work will be conducted on further augmenting the datasets. Although the effects applied were the same in the training, validation and test sets, the implementations used were different in the training set. This leads to a different timbre between the sets that the architecture might not be able to generalise to. In future experiments, we will further augment the dataset using a number of different settings for each effect, as well as different combinations of the effects applied.

5. CONCLUSIONS

In this paper we evaluated how a state of the art algorithm for automatic instrument classification performs when classifying the NSynth dataset and how augmenting this dataset with audio effects commonly used in electronic music production influences its accuracy on both the original and processed versions of the audio. We identify that the accuracy of this algorithm is greatly decreased when tested on sounds where audio effects are applied and see that the augmentation can lead to better classification in unprocessed sounds. We note that the accuracy results provided are preliminary, and do not fully exploit the possibilities of using audio effects for data augmentation in automatic instrument classification. We are currently evaluating how a deeper architecture performs on the same task. Further work includes evaluating how using a bigger variety of effects, with different combinations of parameters, further improves the robustness of the classification algorithm.

6. ACKNOWLEDGMENTS

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement N^{o} 765068, MIP-Frontiers. We thank Matthew Davies for reviewing a draft of this paper and providing helpful feedback.

7. REFERENCES

[1] Frederic Font, Gerard Roma, and Xavier Serra, "Freesound technical demo," in *ACM International Conference on Multimedia (MM'13)*, Barcelona, Spain, 2013, ACM, pp. 411–412, ACM.

[2] Perfecto Herrera-Boyer, Geoffroy Peeters, and Shlomo Dubnov, "Automatic classification of musical instrument sounds," *Journal of New Music Research*, vol. 32, pp. 3–21, 2003.

[3] Masataka Goto, "RWC music database: Popular, classical, and jazz music databases," in *3rd International Society for Music Information Retrieval Conference (ISMIR)*, 2002, pp. 287–288.

[4] Juan J Bosch, Jordi Janer, Ferdinand Fuhrmann, and Perfecto Herrera, "A comparison of sound segregation techniques for predominant instrument recognition in musical audio signals," in *13th International Society for Music Information Retrieval Conference (ISMIR)*, 2012, pp. 559–564.

[5] Oriol Romani Picas, Hector Parra Rodriguez, Dara Dabiri, Hiroshi Tokuda, Wataru Hariya, Koji Oishi, and Xavier Serra, "A real-time system for measuring sound goodness in instrumental sounds," in *Audio Engineering Society Convention 138*, Warsaw, Poland, 2015, p. 9350.

[6] Venkatesh Shenoy Kadandale, "Musical Instrument Recognition in Multi-Instrument Audio Contexts," MSc thesis, Universitat Pompeu Fabra, Oct. 2018.

[7] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436, 2015.

[8] Luis Perez and Jason Wang, "The effectiveness of data augmentation in image classification using deep learning," *arXiv preprint arXiv:1712.04621*, 2017.

[9] Jordi Pons, Olga Slizovskaia, Rong Gong, Emilia Gómez, and Xavier Serra, "Timbre analysis of music audio signals with convolutional neural networks," in *25th European Signal Processing Conference (EUSIPCO)*. IEEE, 2017, pp. 2744–2748.

[10] Yoonchang Han, Jaehun Kim, Kyogu Lee, Yoonchang Han, Jaehun Kim, and Kyogu Lee, "Deep convolutional neural networks for predominant instrument recognition in polyphonic music," *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, vol. 25, no. 1, pp. 208–221, Jan. 2017.

[11] Peter Li, Jiyuan Qian, and Tian Wang, "Automatic instrument recognition in polyphonic music using convolutional neural networks," *arXiv preprint arXiv:1511.05520*, 2015.

[12] Taejin Park and Taejin Lee, "Musical instrument sound classification with deep convolutional neural network using feature fusion approach," *arXiv preprint arXiv:1512.07370*, 2015.

[13] Juhan Nam, Keunwoo Choi, Jongpil Lee, Szu-Yu Chou, and Yi-Hsuan Yang, "Deep learning for audio-based music classification and tagging: Teaching computers to distinguish rock from bach," *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 41–51, Jan 2019.

[14] Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Mohammad Norouzi, Douglas Eck, and Karen Simonyan, "Neural audio synthesis of musical notes with wavenet autoencoders," in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, 2017, pp. 1068–1077.

[15] Brian McFee, Eric J Humphrey, and Juan Pablo Bello, "A software framework for musical data augmentation.," in *16th International Society for Music Information Retrieval Conference (ISMIR)*, 2015, pp. 248–254.

[16] Justin Salamon and Juan Pablo Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 279–283, 2017.

[17] Tom Ko, Vijayaditya Peddinti, Daniel Povey, Michael L Seltzer, and Sanjeev Khudanpur, "A study on data augmentation of reverberant speech for robust speech recognition," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 5220–5224.

[18] Udo Zölzer, *DAFX: Digital Audio Effects*, John Wiley & Sons, 2011.

[19] Joshua D Reiss and Andrew McPherson, *Audio effects: theory, implementation and application*, CRC Press, 2014.

[20] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[21] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[22] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[23] Brian McFee et al., "librosa/librosa: 0.6.3," Feb. 2019.

[24] Robert Ratcliffe, "A proposed typology of sampled material within electronic dance music," *Dancecult: Journal of Electronic Dance Music Culture*, vol. 6, no. 1, pp. 97–122, 2014.

[25] Shruti Sarika Chakraborty and Ranjan Parekh, *Improved Musical Instrument Classification Using Cepstral Coefficients and Neural Networks*, pp. 123–138, Springer Singapore, Singapore, 2018.

[26] Rachel M. Bittner, Justin Salamon, Mike Tierney, Matthias Mauch, Chris Cannam, and Juan Pablo Bello, "MedleyDB: A multitrack dataset for annotation-intensive MIR Research," in *the 15th International Society for Music Information Retrieval Conference (ISMIR)*, 2014.