Audio Engineering Society

Convention e-Brief 23

Presented at the 147th Convention
2019 October 16 – 19, New York

Vibrary: A Consumer-Trainable Music Tagging Utility

Scott H. Hawley[1], Jason Bagley[2], Brett Porter[2], and Daisy Traynham[2]

[1]*Belmont University*
[2]*Art+Logic, Inc.*

Correspondence should be addressed to Scott H. Hawley (`scott.hawley@belmont.edu`)

ABSTRACT

We present the engineering underlying a consumer application to help music industry professionals find audio clips and samples of personal interest within their large audio libraries typically consisting of heterogeneously-labeled clips supplied by various vendors. We enable users to train an indexing system using their own custom tags (e.g., instruments, genres, moods), by means of convolutional neural networks operating on spectrograms. Since the intended users are not data scientists and may not possess the required computational resources (i.e. Graphics Processing Units, GPUs), our primary contributions consist of i) designing an intuitive user experience for a local client application to help users create representative spectrogram datasets, and ii) 'seamless' integration with a cloud-based GPU server for efficient neural network training.

1 Introduction

This paper describes a product developed to meet a need expressed by music composers, producers and supervisors: personalized audio tagging. These music industry professionals typically rely on large quantities (e.g. terabytes) of musical audio "clips" to ply their trade, and rely on music tagging to search through their available sounds. Often these clips are part of packages purchased from vendors who do not follow uniform criteria when generating metadata tags, resulting in heterogeneous, contradictory tags which make the search challenging. Furthermore, these tags are "burned in" and do not change with the user's preference, whereas some users want to create their own tags for personalized descriptive criteria such as mood, feel, timbre, etc.

Rather than being an issue of mere metadata resolution, this use case necessitates a method relying on audio features which may be difficult to quantify on the front-end. Since the work of Lee et al.[1], the increasingly popular method of applying convolutional neural networks (CNNs) to audio spectrograms has demonstrated great success as a flexible and robust method for audio feature selection [2, 3] and subsequent classification. This approach has been applied successfully to supervised classification problems as varied as chord recognition [4], genre [5, 6], tempo [7], and artist [8]. In particular, using a mel-frequency scale for the spectrogram has been shown to be particularly powerful [9]. CNNs can be robust with respect to noisy labels [10], and relatively small datasets.

CNNs are used for inference in consumer products such as the Track Assistant in Neutron 2 [11], but have yet to be widely offered for training. Two barriers to training by end users are their lack of graphics processing units (GPUs) for efficient neural network computation, and the lack of a user-friendly interface for dataset creation.

2 System Design

We overcome such barriers by training in the cloud on servers with GPUs, and provide a GUI-based client to assist users in both dataset creation and "sorting" (i.e. generating tags for) new files. An overview of the system's operation in shown in Figure 1.

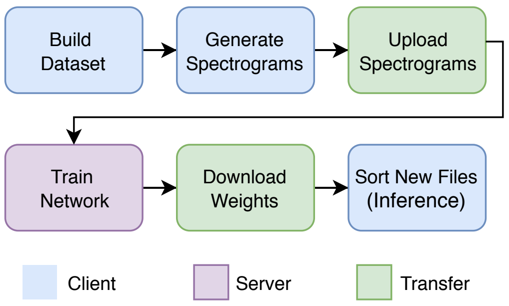
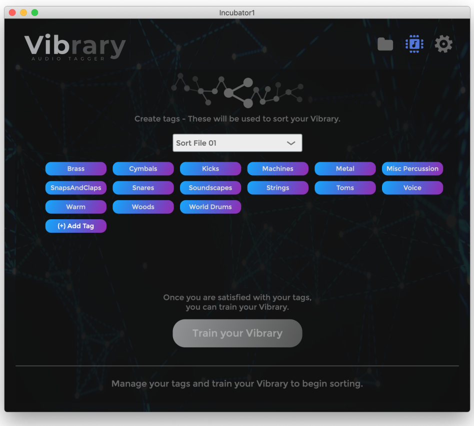


Fig. 1: Overview of Vibrary operation procedure.



Create tags - These will be used to sort your Vibrary.

Once you are satisfied with your tags, you can train your Vibrary.

Manage your tags and train your Vibrary to begin sorting.

Fig. 2: Screenshot for part of the dataset creation process, where tags and corresponding audio examples are added by the user.

2.1 Client

The client is built in C++ using JUCE.[1] An example of the graphical user interface (GUI) can be seen in Figure 2. Initially the user is presented with a "demo mode" with pretrained network weights for 15 different classes of instruments. The user can immediately use this to "sort" large quantities of audio files on their hard drives by running the network in inference mode to associate tags with these files. (This demo model can later be replaced with a newly-trained model, and multiple models may be run in succession for multiple tags.) The client helps users create a training dataset by allowing them to drag and drop (directories of) audio files into the interface, at which point initial tags can be suggested based on metadata. Once the user has supplied tags to associate with these files, the program generates spectrograms and uploads these to the server (see Section 2.2). After the model is trained, the client downloads the model weights in the form of a TensorFlow Saved Model which uses Protobuf as a file format. Then, additional audio files on the users' machine can have tags associated with them – a process we refer to as "sorting" – as the client creates spectrograms for these and runs a local (C++) version of the network in inference mode.

[1]JUCE: `http://juce.com`

2.2 Server and Neural Network

The demo version currently runs on a private server machine, and for full release will use a Dockerized Python build on an Amazon EC2 instance that spins up at a time estimated to coincide with the completion of the data upload to S2 storage.

For the neural network, we employ an existing Open Source code [12], which applies a VGG-style network with 3x3 kernels, Dropout, and ELU activations. Although more sophisticated models are available [13, 14, 15], we find this model to be sufficiently fast and accurate for our use case: Training time on spectrograms for 1 GB of audio typically takes a few minutes or less, and 'accuracy' is determined much more by the user's data curation than by the model type or whether the default of Transfer Learning [6, 16] or random initialization is used.

Thus ours is not an end-to-end model [17], rather we rely on the music domain assumptions inherent in using mel spectrograms. This serves three purposes particular to our use case:

1. **Reduced Upload Size.** The corpus of audio files in our typical datasets so far have a size on the order of gigabytes, which would require long upload times. Producing spectrograms locally reduces this by a factor of 20 to 80. This also allows spectrograms to be archived for re-use with a minimal storage footprint.

2. **Compliance with Licensing.** Uploading raw audio from a commercial package may constitute unauthorized distribution, however our spectrograms lack phase data thus precluding full audio reconstruction.
3. **Performance on 'Small' Datasets.** Although end-to-end models with waveform front-ends can offer the highest accuracy for datasets of millions of instances [18], we expect that users may supply significantly less, for which a spectrogram front-end is likely to offer superior performance [19].

Data augmentation is currently offered in the form of shifting in time and mel-frequency, stretching in time, and mixup. We intend to include SpecAugment [20] soon. A small amount (0.01) of label smoothing is also applied to allow for tolerance to mislabeled examples.

2.3 Considerations for End Users

End users are not data scientists, nor are audio clip packages created with data curation in mind. These have a combined effect on the accuracy of the model. For example, clips designated as 'Snares" by vendors may contain highly "effected" sounds with sustained high frequency content more typical of cymbals. This can result in mixing between "Cymbal" and "Snare" classes, causing the accuracy to suffer.

We supply defaults for user-selectable settings appropriate for the design goal of classifying "short" audio clips for music composition and supervision, in which the sonic characteristics are consistent during the clip, e.g. a default duration of 3 seconds for spectrograms (zero padded), and down-mixing to mono by default. We are currently developing guidelines to suggest the best settings for other use cases.

3 Results

As noted earlier, the accuracy of the classifications will depend on the 'cleanliness' and size of the users dataset. As an aid, we provide users with a sense of how much effort on their part will correlate with classification accuracy, which will also depend on the type of audio being classified. Figure 3 shows average results for classification accuracy on a testing dataset as a function of the number of audio examples used in the training set, for example cases of either 12 audio effect classes[21], 10 genre classes[22], or 9 instrument classes[2].

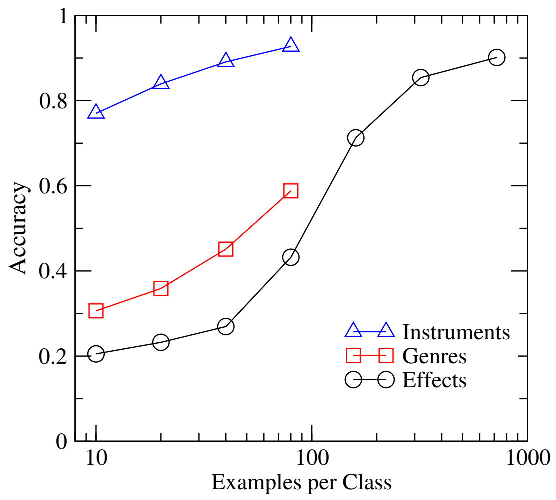


Fig. 3: Testing accuracy as a function of number of example audio files in the training set. This can be used to give users an estimate of how many sample files to supply per class.

4 Summary

Vibrary is a tool to help end users personalize tags of musical audio clips, by means of a client-server model which offloads neural network training to cloud-based GPU servers, with spectrograms generated locally to reduce upload sizes and avoid licensing issues. We are interested to see how users make use of this utility, particularly in terms of training for unanticipated categories. Interested parties should contact the authors to obtain a demonstration copy of the software.

References

[1] Lee, H., Pham, P., Largman, Y., and Ng, A. Y., "Unsupervised Feature Learning for Audio Classification Using Convolutional Deep Belief Networks," in Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pp. 1096–1104, Curran Associates, Inc., 2009.

[2]The authors wish to thank composer Kyle J. Baker for assistance in creating a custom dataset of 9 instrument classes

[2] Humphrey, E. J., Bello, J. P., and LeCun, Y., "Moving beyond feature design: Deep architectures and automatic feature learning in music informatics," in *Proceedings of the 13th International Society for Music Information Retrieval Conference, ISMIR 2012*, pp. 403–408, 2012.

[3] Gwardys, G. and Grzywczak, D., "Deep Image Features in Music Information Retrieval," *International Journal of Electronics and Telecommunications*, 60(4), pp. 321–326, 2014, doi:10.2478/eletel-2014-0042.

[4] Humphrey, E. J. and Bello, J. P., "Rethinking Automatic Chord Recognition with Convolutional Neural Networks," in *2012 11th International Conference on Machine Learning and Applications*, pp. 357–362, 2012.

[5] Defferrard, M., Mohanty, S. P., Carroll, S. F., and Salathé, M., "Learning to Recognize Musical Genre from Audio," in *WWW '18 Companion: The 2018 Web Conference Companion*, 2018.

[6] Kim, J., Won, M., Serra, X., and Liem, C. C. S., "Transfer Learning of Artist Group Factors to Musical Genre Classification," in *Companion Proceedings of the The Web Conference 2018*, WWW '18, pp. 1929–1934, International World Wide Web Conferences Steering Committee, 2018.

[7] Schoneveld, L., "Detecting Music BPM Using Neural Networks," 2016.

[8] Nasrullah, Z. and Zhao, Y., "Music Artist Classification with Convolutional Recurrent Neural Networks," *arXiv*, 2019.

[9] Choi, K., Fazekas, G., and Sandler, M., "Automatic Tagging Using Deep Convolutional Neural Networks," *arXiv*, 2016.

[10] Choi, K., Fazekas, G., Cho, K., and Sandler, M., "The Effects of Noisy Labels on Deep Convolutional Neural Networks for Music Tagging," *arXiv*, 2017.

[11] Izotope, Inc., "Neutron 2 (Track Assistant)," 2017.

[12] Hawley, S. H., "Panotti: A Convolutional Neural Network Classifier For Multichannel Audio Waveforms," 2017, doi:10.5281/zenodo.1275605.

[13] Hershey, S., Chaudhuri, S., Ellis, D. P. W., Gemmeke, J. F., Jansen, A., Moore, R. C., Plakal, M., Platt, D., Saurous, R. A., Seybold, B., Slaney, M., Weiss, R. J., and Wilson, K., "CNN Architectures for Large-Scale Audio Classification," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 131–135, 2017.

[14] Pons, J., Lidy, T., and Serra, X., "Experimenting with Musically Motivated Convolutional Neural Networks," in *2016 14th International Workshop on Content-Based Multimedia Indexing (CBMI)*, pp. 1–6, 2016.

[15] Lee, J., Park, J., Kim, K. L., and Nam, J., "Sample-Level Deep Convolutional Neural Networks for Music Auto-Tagging Using Raw Waveforms," *arXiv*, 2017.

[16] Pons, J., Serrà, J., and Serra, X., "Training Neural Audio Classifiers with Few Data," *arXiv*, 2018.

[17] Dieleman, S. and Schrauwen, B., "End-to-End Learning for Music Audio," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6964–6968, 2014.

[18] McFee, B., Bertin-Mahieux, T., Ellis, D. P., and Lanckriet, G. R., "The million song dataset challenge," in *Proceedings of the 21st International Conference on World Wide Web*, pp. 909–916, ACM, 2012.

[19] Pons, J., Nieto, O., Prockup, M., Schmidt, E. M., Ehmann, A. F., and Serra, X., "End-to-end learning for music audio tagging at scale," *CoRR*, abs/1711.02520, 2017.

[20] Park, D. S., Chan, W., Zhang, Y., Chiu, C.-C., Zoph, B., Cubuk, E. D., and Le, Q. V., "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition," *arXiv*, 2019.

[21] Stein, M., Abeßer, J., Dittmar, C., and Schuller, G., "Automatic Detection of Audio Effects in Guitar and Bass Recordings," volume 1, pp. 522–533, 2010.

[22] Tzanetakis, G. and Cook, P., "Musical Genre Classification of Audio Signals," *IEEE Transactions on Speech and Audio Processing*, 10, pp. 293 – 302, 2002, doi:10.1109/TSA.2002.800560.